Mesoscale Models and Numerical Algorithms for Fracture of Solids

A Dissertation Presented

by

Hongren Wei

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

August 2012

Stony Brook University

The Graduate School

Hongren Wei

We, the dissertation committee for the above candidate for the Doctor of Philosophy degree, hereby recommend acceptance of this dissertation.

Roman Samulyak, Dissertation Advisor
Associate Professor, Dept. of Applied Mathematics and Statistics

James Glimm, Chairperson of Defense
Professor, Dept. of Applied Mathematics and Statistics

Xiangmin Jiao, Committee Member
Associate Professor, Dept. of Applied Mathematics and Statistics

Nikolaos Simos, Outside Member
Senior Scientist, Brookhaven National Laboratory

This dissertation is accepted by the Graduate School

Charles Taber

Interim Dean of the Graduate School

Abstract of the Dissertation

Models and Numerical Algorithms for Fracture of Solids

by

Hongren Wei

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

2012

A new mass conservative mesoscale model for the simulation of fracture of solid materials has been developed. Our representation of solids by spring networks contains two degrees of freedom necessary to match real material properties and exhibits a stable Poisson ratio. The algorithm is based on the energy minimization of the network of triangular springs with critical strain and splitting of overstressed bonds and connected to them nodes ensuring the conservation of mass during the crack evolution. An algorithm to resolve the mesh folding and overlapping for the simulation of compressed materials has been developed by introducing special energy penalty terms. The main emphasis of the research is on the study of brittle fracture but elasto-plastic models for springs have also been developed for the simulation of plastic deformations with limited shear bands. Two regimes of the brittle fracture have been considered: adiabatically slow deformation and breakup and instantaneously fast deformation and the formation and propagation of cracks in stressed materials. Parallel software for the fracture of brittle materials under strain has been developed with the integration of packages TAO and Global Arrays. A Schwartz-type overlapping domain decomposition and the corresponding acceleration techniques have also been studied. Three different visualization techniques have been developed to capture details of fractured zones in 3D. The software has been applied to the simulation of fracture of solids under slow stretching deformations, the rapid disintegration of highly tempered glasses in the phenomenon called the Prince Rupert Drop, and the fracture of thin brittle discs hit by high velocity projectiles. The bifurcation of the fracture dynamics from the growth of the comminuted zone to the propagation of isolated radial cracks, typical for the fracture of glass sheets and thin ceramic plates hit by projectiles, has been reproduced in our numerical experiments and scaling studies involving the change of material properties and projectile velocity have been performed. The fracture model has also been used in a coupled multiscale simulation of the nuclear fuel rod failure within a study of nuclear reactor safety issues.

Contents

List of Figures

Acknowledgments

I gratefully acknowledge my advisor, Professor Roman Samulyak, for the interesting topic suggested and for his guidance and continuous support. I benefited a lot from his broad knowledge and innovative thinking.

It is a great honor to have Professor James Glimm, Professor Xiangmin Jiao and Dr. Nikolaos Simos to be my dissertation committee members. I really appreciate all the valuable suggestions they made to improve and perfect this dissertation.

I would like to thank Professor Xiaolin Li for the guidance and instructions on my study and sharing many precious experiences of him.

Sincere thanks to all my fellow group members for all their discussions and suggestions, especially Gaurish Telang for his help in iso-surface visualization and Kwangmin Yu for sharing his valuable ideas about the data structure design.

Chapter 1

Introduction

Fracture of solid materials including brittle fracture has been studied for decades. Foundations of fracture mechanics was developed during World War I by English aeronautical engineer, A. A. Griffith. Griffith's main motivation to explain the failure of brittle materials was inspired by the fact that the stress required to crack bulk glass is only 1/100 of the stress required to break the atomic bonds of it. In addition, Griffith conducted some of his own experiments on glass fibers and concluded that the fracture stress increases as the fiber diameter decreases. Griffith developed a theory aimed to explain these contradictions [1]. He suggested that the low fracture strength observed in experiments, as well as the size-dependence of strength, was due to the presence of microscopic flaws in the bulk material.

Griffith created an artificial flaw in his experimental specimens to prove his hypothesis. The artificial flaw was much larger than other flaws in a specimen. The experiments showed that the product of the square root of the flaw size a and the fracture stress σ_f was almost constant, which is formulated by:

$$\sigma_f \sqrt{a} \approx C \tag{1.1}$$

Linear elasticity theory could not explain this relation since it predicts that the stress and the strain at the tip of a sharp flaw in a linear elastic material is infinite. To avoid the problem, a thermodynamic approach was proposed by Griffith to explain the relation that he observed.

Further, through the experiments he performed, Griffith found that

$$C = \sqrt{\frac{2E\gamma}{\pi}} \tag{1.2}$$

where E is the Young's modulus of the material and γ is the surface energy density of the material, and gives excellent agreement of Griffith's predicted fracture stress with experimental results.

The relation $\sigma_y \sqrt{a} \approx C$ still holds for ductile materials such as steel, but the surface energy γ predicted by Griffith's theory is usually much larger than the real value. In 1940s, a research group working under G. R. Irwin at the U.S. Naval Research Laboratory (NRL) found that plasticity plays a significant role in the fracture of ductile materials and introduced modifications to Griffith's theory [2].

$$C = \sqrt{\frac{2EG}{\pi}} \tag{1.3}$$

where $G = 2\gamma + G_p$. The plastic dissipation term will dominates when the material is ductile and $G_p \approx G$. When the material is brittle G_p is neglectable. But there was a problem to the revised model because naval materials are not perfectly elastic but show significant plasticity at the tip of a crack. One basic assumption in Irwin's fracture model is that the size of the plastic zone is small compared to the crack length. However, this assumption is very limited for certain types of failure in structural steels though such steels can be prone to brittle fracture.

The processes of material fracture exhibits a rich phenomenology extending over a wide range of scale from the atomic level to the system scale. While it remains difficult to describe the complex behavior on the fundamental level, several computational approaches have been developed to describe the fine structure of the fracture zone on the phenomenological level.

Solid fracture is similar to numerous other processes leading to the formation of complex patterns such as dielectric breakdown, fluid-fluid displacement in porous medium, Hele-Shaw cell, phase transition (solidification, crystal growth) etc. Meakin draws a connection of the fracture mechanics to the diffusion-limited aggregation(DLA) model of Witten and Sander [3,4] in which the growth of probabilities in a random growth process are controlled by a scalar field obeying the Laplace equation. The DLA model also provides a basis for understanding several other random pattern formation processes where the growth process is controlled by a field obeying the Laplace equation [9,10,11,12]. One of the most important characteristics of structures generated by DLA models is their fractal geometry [15]. A fractal dimensionality of about 1.71 for 2D and about 2.50 for 3D [16,17,18] have been generated by the DLA model. These values agree with the results achieved in related experiments. Numerical simulations have been used widely to analyze different aspects of mechanical failure and most of these simulations are performed by molecular dynamics approaches [23,24,25,26].

We would like to mention another interesting approach of this class to the description of brittle fracture by failure waves [31]. The the model has similar features with a simplified theory of slow combustion. Therefore the brittle failure is considered as a phase transition-type process. The internal wave structure of the wave front is ignored, and the failure front is treated as a mathematical surface. Given properties of the damaged and undamaged phases, a system of partial differential equations for the evolution of each phase and jump conditions can be derived using the balance equations. Such a system describes the propagation of failure waves and their morphological instabilities. The complex structure of fracture zones are described as a geometrically complex interface between the damaged and undamaged phases due to instabilities of the failure wave front. This theory contains several deficiencies. One of them is the need to know material properties of the fractured zone which are difficult or impossible to obtain reliably from experiments or calculate theoretically. But they use a set of parameters for material properties of the damaged zone is a big simplification due to strong non-uniformity of damaged zones. Another simplification is the simple connectivity assumption of the wave front: the description of the failure zone as a result of the evolution of a geometrically complex but simply connected wave front. This however can be improved by 3D numerical simulations based on the failure wave theory that starts from realistic initial distribution of stresses. In such numerical simulations, numerous failure wave fronts would interact leading to more complex fracture zones. In contrast, our model does not require properties of the damaged phase and leads to the growth of complex fracture zones and their transition to a small number of distinct cracks using more fundamental principles.

Peridynamics [44,45,46,47] is a meshfree method for continuum solid mechanics to simulate discontinuities. The material is represented by a collection of particles, and each particle interacts with other particles that are in a specific neighborhood, which is called horizon.The horizon is usually 2-3 lattice spacings. The interaction between two points is connected by a linkage bond which holds bond force density as well as a predefined damage state function. When the damage state function is positive, the bond break thus the interaction between those two points get lost. Peridynamics solves integral equations instead of partial differential equations in classical continuum mechanics.

Another method in fracture simulation is the extended finite element method(XFEM) [61]. XFEM is a technique to simulate the crack discontinuities that not necessarily conform to the finite element mesh. A discontinuous function and crack-tip asymptotic functions are added to the regular finite element method. The crack interior is represented by a discontinuous function while the crack-tip by the asymptotic crack-tip functions. XFEM retains the advantages of FEM and added new advantages, such as avoids the need to remesh as the crack grows, so it is widely used in fracture simulation since its debut in 1999. Initially it was used only to model one crack under tensile stress, but later been developed with capabilities to handle more complex crack bifurcation and compression stress [62,63]. But it is fairly difficult for XFEM to resolve complicated discontinuity patterns [62] and the free flow of cracked fragments. Our model has the advantages in this aspect.

Several phenomenological models have been developed using a energy minimization of spring networks. In his two-dimensional model for crack propagation [32] , Meakin used a 2D triangular network in which each bond is represented by a linear spring with critical tension. The location of nodes in deformed solids is found by the energy minimization and bonds which exceed the critical strain are removed from the system. However, different types of artificial techniques are employed to control the crack surface propagation. Beale used a similar model [33] but randomly added some percentage of initial defects to the system and studies the distribution of failure stress. But the bond-breaking mechanism both of them used is removing the overstressed bond which will generate void regions and lead to the loss of mass. When the material is under compression, the mass loss will be large. Several other approaches based on force methods have demonstrated satisfactory simulations of cracked zones in simple setups.

The main deficiencies of the described previous simulation models based on the energy minimization of spring networks are

- Algorithms applicable only to stretched solids. Unphysical mesh folding occurs if these models are applied to solids under compression forces.

- Non-conservative bond removal algorithm which leads to the loss of mass. This issue is less important if small number of isolated cracks developed in stretched solids. But it prevents compressed solids, in particularly the simulation of comminuted fracture zones.
- Inability to reproduce complex patterns occurring in brittle fracture processes (for instance a comminuted zone and a series of radial cracks in solid plates hit by high velocity projectiles).
- Only 2D dimensional models
- Serial codes for running on single-processor computers

In this thesis, we describe new mathematical models, numerical algorithms, parallel software, and numerical simulations that resolve all deficiencies described above. The structure of the thesis is as following. In chapter 2, we will introduce the main notions and equations of theory of elasticity and establish the mathematical model for brittle fracture of solids in chapter 3. In chapter 4, the main components of the numerical algorithms and software mesh generation, energy minimization, mass-conservative crack propagation algorithm, parallelization and visualization will be introduced. In chapter 5, we will show several numerical results, validation and applications. The thesis will conclude with future work in chapter 6.

Chapter 2

Main Notions and Equations of Theory of Elasticity

Before writing the equations of the theory of elasticity, we give a brief introduction to the parameters that are used frequently hereafter in this article.

Stress is a parameter to describe the force applied per unit area. Suppose in the Cartesian coordinate system, the vector sum of all the forces acted on the material is zero. Take a slice orthogonal to the x-axis and let a small area in this slice to be ΔA_x. Define the total forces acting on ΔA_x to be:

$$\Delta F = \Delta F_x \hat{i} + \Delta F_y \hat{j} + \Delta F_z \hat{k} \tag{2.1}$$

The following parameters can be defined:

$$\sigma_{xx} = \lim_{\Delta A_x \to 0} \frac{\Delta F_x}{\Delta A_x}, \sigma_{xy} = \lim_{\Delta A_x \to 0} \frac{\Delta F_y}{\Delta A_x}, \sigma_{xz} = \lim_{\Delta A_x \to 0} \frac{\Delta F_z}{\Delta A_x} \tag{2.2}$$

The subscripts i and j in σ_{ij} are the directions of plane and force respectively. Similarly, the stresses in the plane orthogonal to the y and z axes can be derived:

$$\sigma_{yx} = \lim_{\Delta A_y \to 0} \frac{\Delta F_x}{\Delta A_y}, \sigma_{yy} = \lim_{\Delta A_y \to 0} \frac{\Delta F_y}{\Delta A_y}, \sigma_{yz} = \lim_{\Delta A_y \to 0} \frac{\Delta F_z}{\Delta A_y} \tag{2.3}$$

$$\sigma_{zx} = \lim_{\Delta A_z \to 0} \frac{\Delta F_x}{\Delta A_z}, \sigma_{zy} = \lim_{\Delta A_z \to 0} \frac{\Delta F_y}{\Delta A_z}, \sigma_{zz} = \lim_{\Delta A_z \to 0} \frac{\Delta F_z}{\Delta A_z} \tag{2.4}$$

Those quantities form the following stress tensor matrix:

$$\sigma = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix} \tag{2.5}$$

Stresses are categorized into two types: normal stress, which has $i = j$ in σ_{ij} and shear stress, which has $i \neq j$ in σ_{ij}. The one used in this article is the normal stress, which can also be defined approximately as the normal force applied per unit area of material section.

$$\sigma = F_n / A \tag{2.6}$$

The material under deformation bears an internal restoring force, and the deformation is called strain. Assume there are a pair of points $\vec{\mathbf{x}}$ and $\vec{\mathbf{x}} + \mathbf{d}\vec{\mathbf{x}}$, which moved to $x + u(x)$ and $x + dx + u(x + dx)$ respectively after the deformation. The distance between the deformed points is:

$$\sum_i (dx_i + u_i(x + dx) - u_i(x))^2 \tag{2.7}$$

Assuming dx is small, the following can be derived by a Taylor expansion about the point x

$$\sum_i (dx_i + \sum_j \frac{\partial u_i}{\partial x_j} dx_j)^2 = \sum_i dx_i^2 + 2\sum_{i,j} dx_i \frac{\partial u_i}{\partial x_j} dx_j + \sum_{i,j,k} \frac{\partial u_i}{\partial x_j} dx_j \frac{\partial u_i}{\partial x_k} dx_k$$

$$= \sum_{i,j} dx_i \left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) dx_j + \sum_k \frac{\partial u_k}{\partial x_i} \frac{\partial u_k}{\partial x_j} \right] dx_j$$

$$= 2\sum_{i,j} dx_i \epsilon_{ij} dx_j$$

Where ϵ_{ij} is the components of the strain tensor:

$$\epsilon_{ij} = \frac{1}{2} \left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \sum_k \frac{\partial u_k}{\partial x_i} \frac{u_k}{x_j} \right]$$

Assuming $\frac{\partial u_k}{\partial x_i} \ll 1$, the second order can be neglected, and we can have

$$\epsilon_{ij} = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) \tag{2.8}$$

And we can have the strain tensor matrix:

$$\epsilon = \begin{pmatrix} \epsilon_{xx} & \epsilon_{xy} & \epsilon_{xz} \\ \epsilon_{yx} & \epsilon_{yy} & \epsilon_{yz} \\ \epsilon_{zx} & \epsilon_{zy} & \epsilon_{zz} \end{pmatrix} \tag{2.9}$$

Note that when $i = j$, $\epsilon_{ii} = \frac{\partial u_i}{\partial x_i}$. This is the normal strain and all other components are called shear stain.

Intuitively, strain is the deformation per unit length along the direction of stress

$$\epsilon = \frac{(L - L_0)}{L} \tag{2.10}$$

ϵ is positive if the material is stretched, negative if compressed.

The Young's modulus E is used to describe the tensile stiffness of elastic solid materials. It is defined as the ratio of tensile stress σ over the tensile strain ϵ in the direction,

$$E = \sigma / \epsilon \tag{2.11}$$

Poisson effect is the phenomena that when a solid material is compressed in one direction, it usually expands in other two directions, and Poisson's ratio ν is a quantitative measure of Poisson effect. For isotropic material, if the it is compressed along X axis,

$$\nu = -\frac{\epsilon_y}{\epsilon_x} = -\frac{\epsilon_z}{\epsilon_x} \tag{2.12}$$

Similarly, the following are also correct:

$$\nu = -\frac{\epsilon_x}{\epsilon_y} = -\frac{\epsilon_z}{\epsilon_y}, \nu = -\frac{\epsilon_y}{\epsilon_z} = -\frac{\epsilon_x}{\epsilon_z} \tag{2.13}$$

Another important measure in solid material description is the stress-strain curve, which shows how the stress and stain are correlated. Usually strain is taken as the abscissa, stress as the ordinate.

The curve in the top of the picture represents typical ductile materials, including gold, lead etc., and the bottom one represents typical brittle materials, including glass, concrete, carbon fiber etc.. For both types of materials, we can see that when the deformation(strain) is smaller than a specific threshold, the stress-strain curve can be

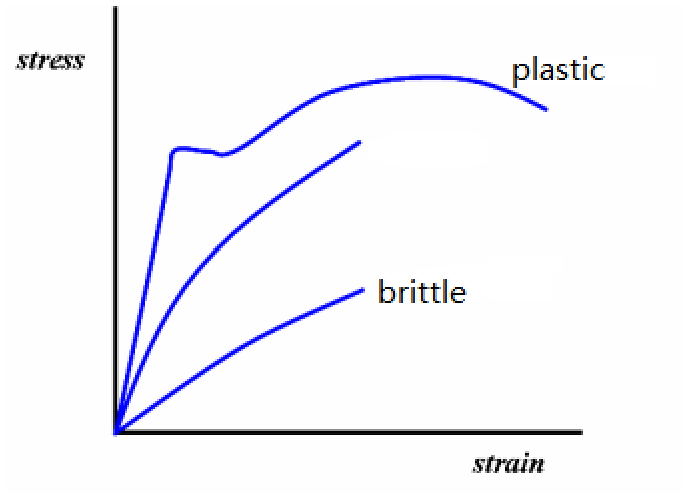


Figure 2.1: Typical stress-strain curves of plastic and brittle materials

treated as linear.

For linear isotropic materials subjected only to normal forces, the parameters mentioned above are coupled by following three equations:

$$\begin{aligned}
\epsilon_x &= \frac{1}{E}[\sigma_x - \nu(\sigma_y + \sigma_z)] \\
\epsilon_y &= \frac{1}{E}[\sigma_y - \nu(\sigma_x + \sigma_z)] \\
\epsilon_z &= \frac{1}{E}[\sigma_z - \nu(\sigma_x + \sigma_y)]
\end{aligned} \tag{2.14}$$

Chapter 3

Mathematical Models for Brittle Fracture of Solids

3.1 Model for Elastic Media

In our model, an elastic medium is represented by a network of springs that satisfy a specific stress-strain relationship. Under the action of external forces, the locations of network nodes are derived by the minimization of the total system energy function. Since the deformation dynamics is not resolved in real time, the springs do not contain mass. Such an approach allows us to eliminate unwanted fast waves in the system such as the sound waves and use relatively large time steps in the process of deformation of solid objects.

Our approach also eliminates the need for an artificial damper term often used in the description of solid dynamics with a mass-spring network. The damper term is relevant and important only for a dynamic spring model which is evolved by calculating the force and acceleration. One of the main features of our model is the avoidance of dynamics by the energy minimization. Consider a simple example of a harmonic spring. Without damping, it will continue vibrating if the initial state is off the equilibrium. However the energy minimization gives the unique and stable equilibrium state corresponding to the given external force.

The difference between the dynamic description of solids by either using differential equations of elasticity[34] of the dynamic mass-spring model and our energy minimization approach is similar to the Lagrangian and Hamiltonian formalisms for classical mechanics. In order to move the system from the initial state to the final state, one can solve the dynamic Hamiltonian or Lagrangian equations[35]. The solution describes the temporal evolution of the system along the correct trajectory in the phase space. Alternatively, one can find the correct trajectory among all possible trajectories, the one along which the functional of action $S = \int L(q, \dot{q}, t), dt$ is minimal, $\delta S = 0$, where $L(q, \dot{q}, t)$ is the Lagrangian function of generalized coordinates q, velocities $\dot{q}$, and time t.

The elastic spring and mass spring models are widely used in the field of computer graphics [37,38,39]. However the important question of the ability of a spring network to accurately represent elastic properties of a real material needs additional clarifications. In particular, we need to discuss the possibility to choose springs constants in such a way that a set of two independent material moduli out of Bulk modulus (K), Young's modulus (E), Lame's first parameter (λ), Shear modulus (G), Poisson's ratio (ν), P-wave modulus (M), are satisfied.

Meakin [32] used a two-dimensional network of elastic springs to represent the material. Each spring has an initial equilibrium length l_0 and a spring constant k_{ij}. $k_{ij} = k$ if nodes i, j are joined, $k_{ij} = 0$ otherwise. The total energy of the network is

$$E = \frac{1}{2} \sum_{ij} k_{ij} (l_{ij} - l_0)^2 \tag{3.1}$$

But this model does not have enough freedom to reproduce two independently experimentally observable moduli: the Poisson's ratio and Young's modulus. To improve it, Gelder [37] proposed a modification of the spring constant k which is dependent on Poisson's ratio and Young's modulus and made it suitable to the irregular triangular meshes.

For a mesh edge c, let $|c|$ be its length. The spring constant k_c is defined by the equation $f = k_c\Delta|c|$, or $Energy = 0.5k_c(\Delta|c|)^2$, where f is the stretching force, $\Delta|c|$ is the change in length in response of f, and $Energy$ is the potential energy held in the spring. All previous papers presumed that the spring constants are equal. He claimed that this presumption is the reason for the quite noticeable distortions of the meshes.

He derived a formula for varying the spring constants of an edge according to the geometry of the triangles incident on that edge as following:

$$k_c = (\frac{E_2}{1+\nu})\frac{\sum_e area(T_e)}{|c|^2} + (\frac{E_2\nu}{1-\nu^2})\frac{|a|^2+|b|^2+|c|^2}{8area(T_e)} \tag{3.2}$$

But all his implemented experimental validations are based on $\nu = 0$, which is simplified to the following formula,

$$k_c = \frac{E_2\sum_e area(T_e)}{|c|^2} \tag{3.3}$$

where the sum is over the triangles T_e incident upon edge c and the coefficient E_2 is the two-dimensional Young's modulus of the membrane to be simulated.

However this method was disproved by Baudet in [38]. He tested the model with materials whose $\nu \geq 0$ on a square object meshed by four symmetrical Van Gelder triangles. The input Young's modulus E_{VG} and input Poisson's ratio ν_{VG}. The spring constants were calculated by equation mentioned above, and then calculate the elastic parameters E, ν, which should be consistent with the input parameters E_{VG}, ν_{VG}.

But the test showed an error of 25% on Young's modulus when input Poisson's ratio $\nu_{VG} = 0$, and the errors increased as input Poisson's ratio increased, where when $\nu_{VG} = 0.5$ the error is as large as 44%. Further more, he derived that the Young's modulus depends on the Poisson's ratio, although the two parameters should be totally independent in linear, isotropic and homogeneous materials.

Herve Delingette [39] proposed a model with the extra angular stiffness terms for modeling nonlinear membranes, with the following relationship:

$$Energy = \sum_{i=1}^{3} \frac{\kappa_i^{T_P}}{4}(l_i^2 - L_i^2)^2 + \sum_i \frac{c_k^{T_P}}{2}(l_i^2 - L_i^2)(l_j^2 - L_j^2) \tag{3.4}$$

where

$$\kappa_i^{T_P} = \frac{2cot^2\alpha_i(\lambda+\mu)+\mu}{16A_P} = \frac{E(2cot^2\alpha_i + 1 - \nu)}{16(1-\nu^2)A_P}$$

$$c_k^{T_P} = \frac{2cot\alpha_i cot\alpha_j(\lambda+\mu)-\mu}{16A_P} = \frac{E(2cot\alpha_i cot\alpha_j + \nu - 1)}{16(1-\nu^2)A_P}$$

In our study, we use general triangular spring meshes and energy minimization methods, with improvement of mass-conservative breaking mechanism and quasi mesh-overlapping resistant methods.

In the next section, we will demonstrate the applicability of our spring model by calculating Young's modulus and the Poisson's ratio.

3.2 Material properties of the spring model and their validation

Our model is based on triangular finite element mesh which is used to represent the material and construct the computational algorithms. The material stiffness is

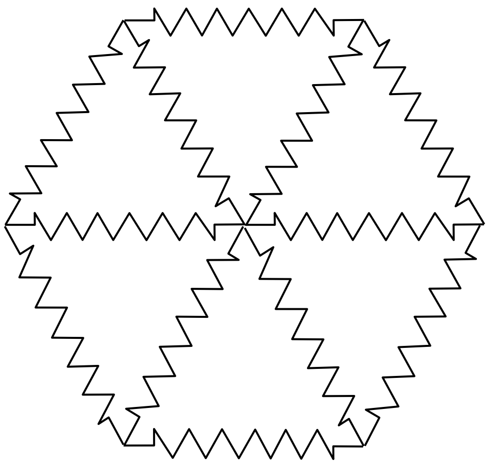


Figure 3.1: spring network represented by unstructured mesh

composed by edge stiffness and element stiffness. The edges in the mesh are treated as elastic springs and thus the edge stiffness is constructed. The element stiffness is constructed by introducing virtual bonds between the incenters of each neighbored pair of triangles, and those virtual bonds are treated the same to the edges except the value of the spring constants. The details are stated below.

A linear elastic spring is a spring that the deviation $L - L_0$ from the equilibrium state is proportional to the tension F in the spring, $F = k_{bond}(L - L_0)$, from which the energy E_i held in the spring can be derived as

$$E = \frac{1}{2}k_{bond}(L - L_0)^2 \tag{3.5}$$

where the coefficient k_{bond} is the elastic coefficient or spring constant used to describe the stiffness of the spring.

The spring constant of bond k_{bond} is selected according to Young's modulus E which satisfies $k \propto E$. Physically, the selection of k_{bond} doesn't affect the equilibrium locations of the mesh nodes in the model. It only affects the numerical stability of the optimization algorithms. However, if k_{bond} is changed, as a result, the tension of the

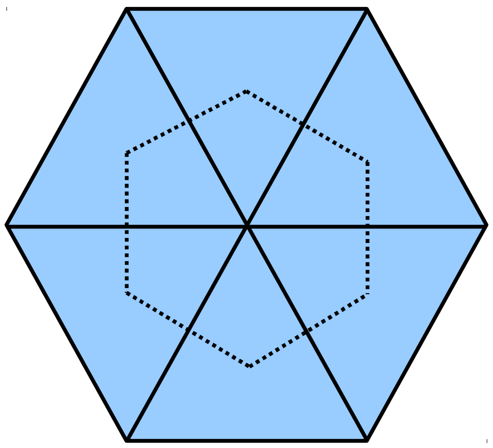


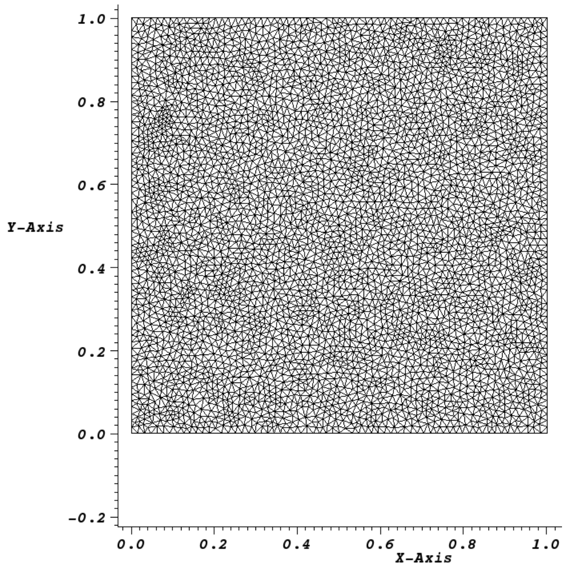
Figure 3.2: Illustration of edge bonds and virtual element bonds. The solid lines represent edge bond, and the dashed lines represent virtual bonds between incenters.

bond and the force applied to each mesh node will be changed. So we can put an adjustable constant c before k_{bond} to make the system consistent with different mesh sizes.

And the equilibrium length d of the virtual bonds among the triangle incenters are also been initialized at the beginning of the simulation. Similarly, we have

$$E_{ele} = \frac{1}{2}k_{ele}(d - d_0)^2 \tag{3.6}$$

As we mentioned before, k_{bond} can only change the stiffness of the material, but with this additional energy constraint we have more freedom to control the Poisson's ratio. The Poisson's ratio is estimated within the small deformation. From our results, we show that when the deformation is controlled under 10%, with $k_{bond} = 1.0$ and $k_{ele} = 1.0$, the result is almost linearly increasing from 0.298 to 0.301. If keep $k_{bond} = 1.0$ constantly, and change k_{ele}, we find that the bigger k_{ele} is, the bigger Poisson's ratio is, but it never exceeds 0.44. Our Poisson's ratio estimation is established as following: Initially, we discretized a unit 1×1 square into a triangular mesh, and then keep the top and



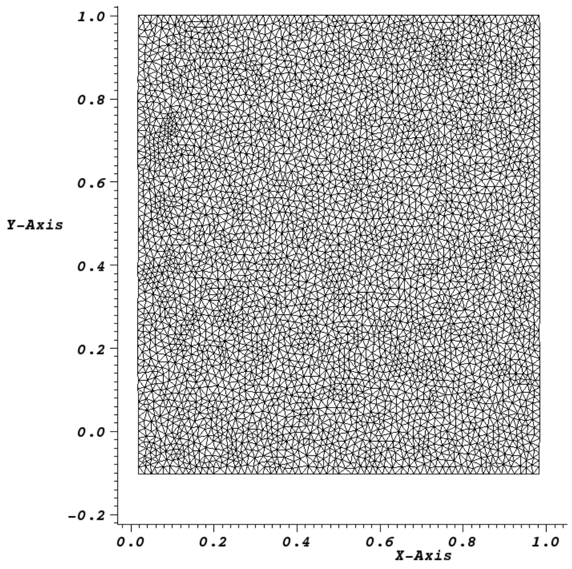


Figure 3.3: Poisson's ratio calculation from a unit square mesh. Left: Initial mesh; Right: one intermediate frame

bottom fixed vertically but can move horizontally. The bottom side moves down δy each step, At each step N, we calculate the current average width and height of the material, noted by $1 - \Delta x$ and $1 + \Delta y$ respectively. Poisson's ratio at this step can be given by,

$$\nu = \frac{\Delta y}{\Delta x} \tag{3.7}$$

For each k_{ele} the relationship of Poisson's ratio ν changes with Δy is shown by a curve in the picture. $k_{ele} = 0, 0.01, 0.1, 1, 10, 100, 500, 5000$ are tested. Here is the Poisson's ratio with respect to different k_{elei}

From the picture, we know that when $k \geq 1$, the Poisson's ratio ν changes very little when the deformation increases. And even when k is very large (5000 in our test), ν is still less than 0.44, which is reasonable because typically $\nu < 0.5$. We can see that the Poisson's ratio in our model is adjustable. In later simulations, $k = 1$, or Poisson's

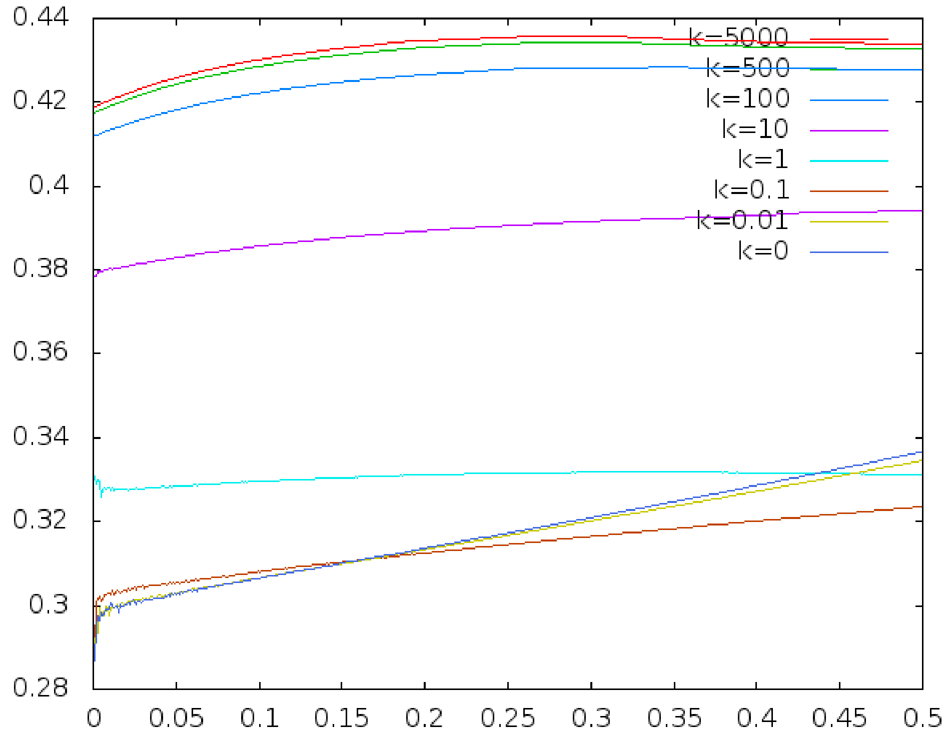


Figure 3.4: Poisson's ratio with $k_{bond} = 1$ and different k_{ele} which is k in the picture.

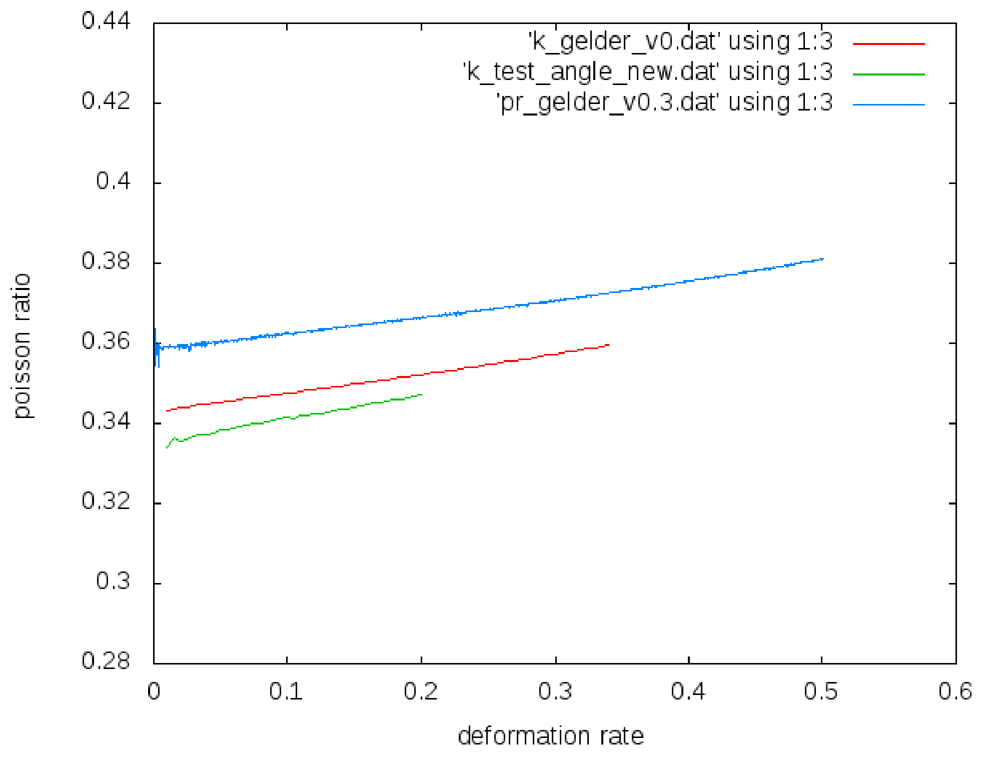


Figure 3.5: Tested Poisson's ratio in Gelder's model and Delinette's model.

ratio $\nu = 0.328$ is used.

The Poisson's ratio in Gelder's model and Delinette's model have also been tested. In both models, ν is an explicit input parameter. We tested the output Poisson's ratio using the same setup as above. The test found that, in Gelder's model, when the input $\nu = 0$, the value used in all the verification simulations in his paper, the output ν is between 0.343 and 0.360 with deformation rate 1% to 34%. When input $\nu = 0.3$, the output ν is between 0.364 to 0.381 which means the error is greater than 20%. In Delinette's model, where angular stiffness was introduced, with a input $\nu = 0.3$ the output is between 0.334 and 0.347 with a deformation rate up to 20%. The Poisson's ratio in these two models are also adjustable since they take it as an explicit parameter, but the real output Poisson's ratio is quite far to the input value and it significantly changes when the deformation increases.

3.3 Crack Formation and Propagation Models

There are two deformation regimes: fast deformation and slow deformation used in the simulation. In the slow deformation regime, we assume that the speed of the crack propagation is slower than the speed of deformation. In the algorithm, after moving the boundary for one step, we apply the mesh manipulation algorithm to process the mesh where there is overstressed bond followed by the energy minimization algorithm to fully minimize the energy after that. The application of a small strain, energy minimization and mesh manipulation algorithm are repeated until the total deformation reaches the desired value. The algorithm is schematically shown in figure 3.6.

In the fast deformation regime, the speed of the crack propagation is assumed to be much faster than the speed of the deformation. Therefore the deformation and energy minimization steps are applied to the system without the mesh manipulation algorithm until the desired total strain is reached. As a result of this strain, a relatively large portion of the solid body might be overstressed. Then the mesh manipulation algorithm is applied to split the overstressed bonds. And then do the energy minimization and mesh manipulation in the way mentioned above iteratively until no bond is overstressed. The theoretical foundation of this algorithm will be described later.

3.4 Plastic Deformations

As is mentioned in chapter 2, elastic deformation will create an internal force which is in the opposite direction of the external force that generates the deformation, and the bigger the deformation, the bigger the internal force. If the external force is released, the material goes back to its original shape. In the process of plastic deformation, the internal force almost remains the same while the deformation increases. The deformation created by plastic deformation is irreversible. The transition of material behaviors

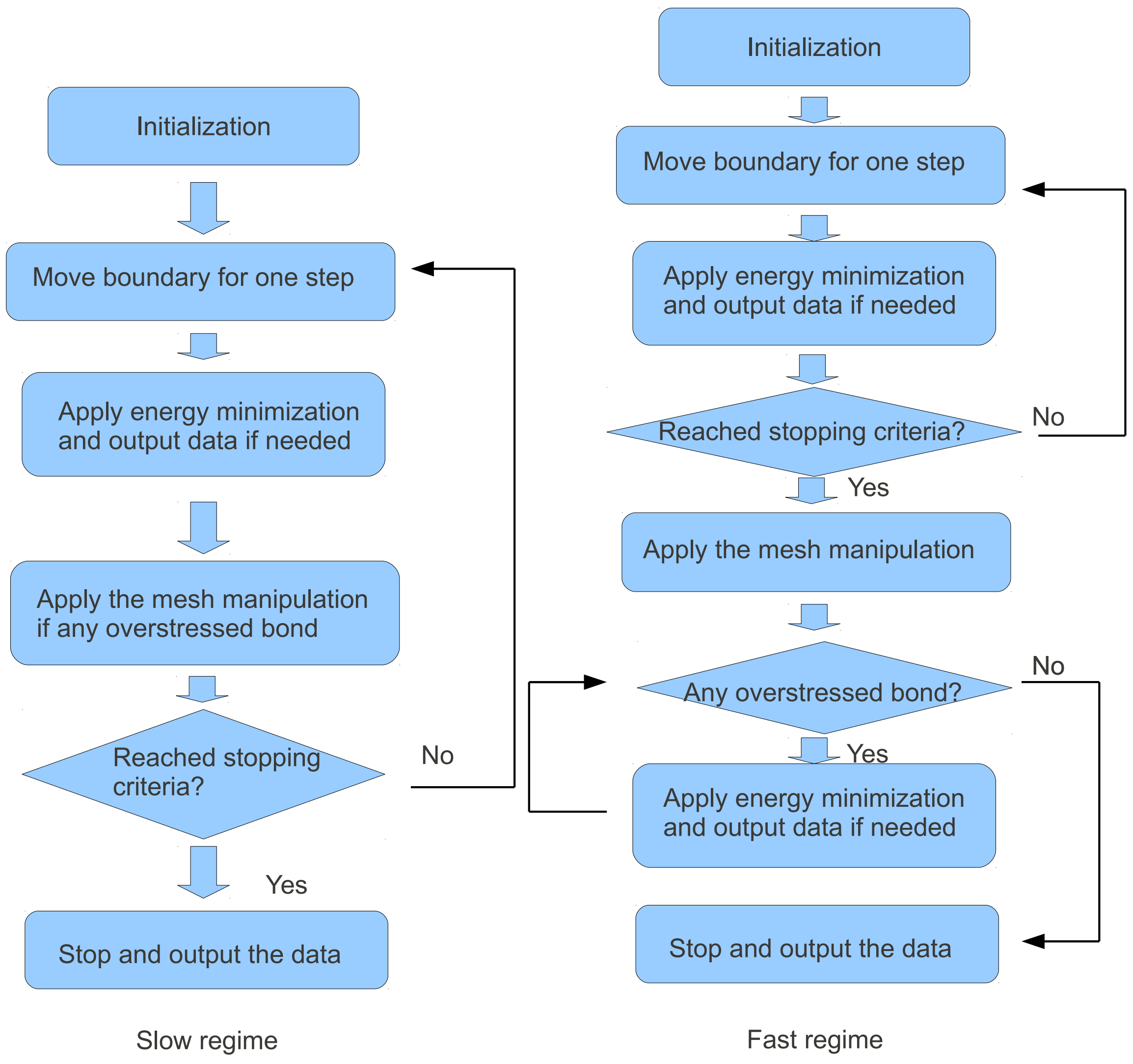

Initialization
Move boundary for one step
Apply energy minimization and output data if needed
Apply the mesh manipulation if any overstressed bond
Reached stopping criteria?
No
Yes
Stop and output the data
Slow regime
Initialization
Move boundary for one step
Apply energy minimization and output data if needed
Reached stopping criteria?
No
Yes
Apply the mesh manipulation
Any overstressed bond?
No
Yes
Apply energy minimization and output data if needed
Stop and output the data
Fast regime

Figure 3.6: flowcharts of slow regime and fast regime

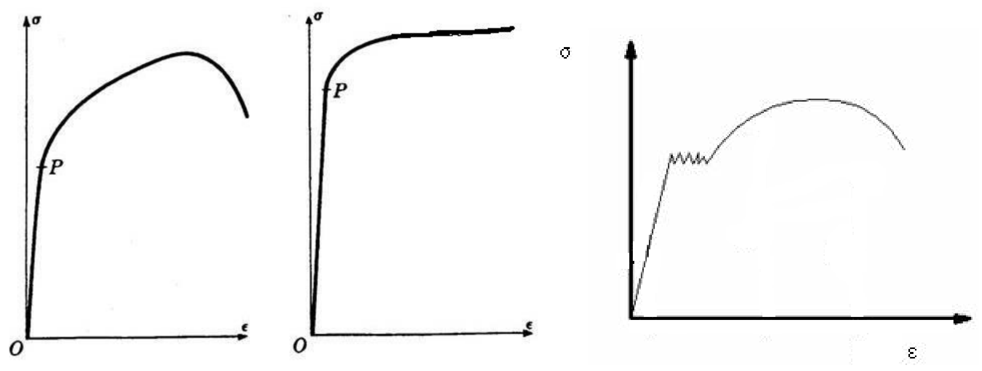


Figure 3.7: stress-strain curves of different plastic materials.

from elasticity to plasticity is called yield.

In the fracture of plastic materials there is an apparent linear elasticity region when the deformation is small. When the deformation grows further, the material will show the properties of plasticity, as shown in figure 3.7. Currently, our model is only capable of handling the materials with positive stress-strain curve slope, e.g. the curve in the middle. For the curve that has non-positive slope, e.g. the one in the left and right, one stress value may correspond to two strain states. Thus the problem is ill-posed and we cannot guarantee the iterative energy minimization method will converge to one specific state out of the two. So we assume the stress-strain curve is non-negative for all the plastic material simulations later. Another difficulty in plastic deformation is that, in practice, when the material fractures there is an amount of permanent deformation after the load is released, which means the deformation will not fully go back to initial equilibrium state as the elastic deformations.

Chapter 4

Numerical Algorithms and Software

4.1 Mesh Generation

The 2D finite element meshes used in the simulations are generated by Triangle [36], a package developed by J. Shewchuk.

Triangle is 2D mesh generator that can produce Delaunay triangulations, constrained Delaunay triangulations, conforming Delaunay triangulations, Voronoi diagrams, and high-quality triangular meshes or refine a previous generated mesh. For the high-quality triangular meshes, user have the control of the minimum angle allowed in all triangles and the maximum triangle area. The output files include, but user have the right to suppress or activate, vertex list, edge list, face list, a list of triangles neighboring each triangle and Voronoi diagrams and optional boundary markers for vertex and edge lists. 4.1 is a sample mesh generated by Triangle.

The 3D tetrahedral meshes are generated by TetGen, a package created by Si Hang of Weierstrass Institute for Applied Analysis and Stochastics, Germany.

TetGen [56] is a package that can generate tetrahedral meshes of any 3D polyhedral domains. TetGen generates exact constrained Delaunay tetrahedralizations, boundary

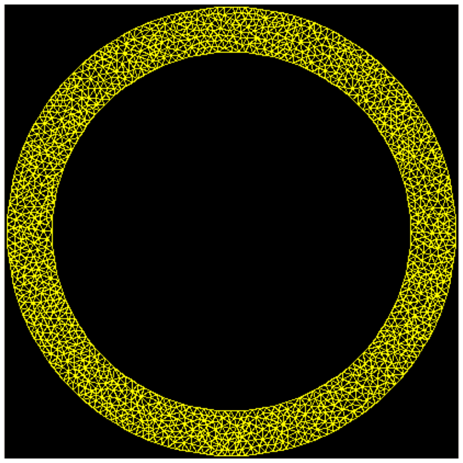


Figure 4.1: A sample 2D meshes generated by Triangle.

conforming Delaunay meshes, Voronoi partitions as well as mesh refinement/coarsening and surface mesh refinement.

The mesh quality is controlled by the radius-edge ratio $Q = R/L$, where R is the radius of the circumsphere of the tetrahedron, and L is the length of the shortest edge of the tetrahedron.

For well-shaped tetrahedron, this measurement is effective where the ratio is usually small, and for most badly-shaped ones, this ratio is large. Even though it cannot guarantee that tetrahedrons with small radius-edge ratios are always in good quality, it is the most natural and elegant measure for analyzing Delaunay refinement algorithms.

Figure 4.2 is a sample 3D tetrahedral mesh generated by TetGen.

4.2 Energy Minimization

Most of the energy minimization is performed by calling the package Toolkit for Advanced Optimization(TAO) [52], a package developed by Argonne National Laboratory. TAO is suitable for both single-processor and multiple-processor architectures and the current version of TAO has algorithms for unconstrained and bond-constrained

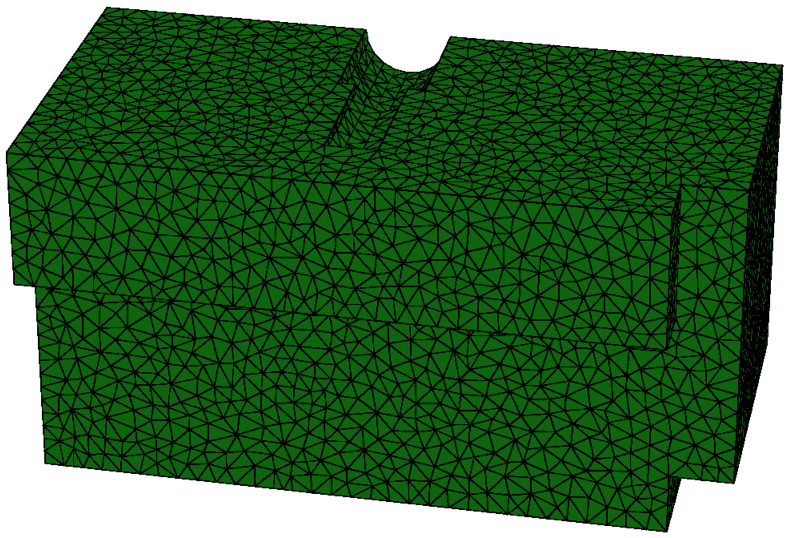


Figure 4.2: A sample 3D mesh generated by TetGen

optimization. The bond-constrained optimization is the problem to minimize,

$$F(\vec{x}) w.r.t. \vec{c_1} \leq \vec{x} \leq \vec{c_2} \tag{4.1}$$

where $\vec{c_1}$ and $\vec{c_1}$ are two constant vectors.

But in our objective function, the constraints are inequalities and/or equalities:

$$F(\vec{x}) w.r.t. g(\vec{x}) \leq c_1 and/or h(\vec{x}) = c_2 \tag{4.2}$$

where c_1 and c_2 are two constant scalars.

In this case, which TAO cannot handle directly, we use an algorithm called penalty method. A constrained optimization is given by Minimize $F(\vec{x}), w.r.t. \mathbf{x} \in \mathbf{S}$ where $F(\vec{x})$ is a continuous function on R^n, and S is a set of constraints in R^n. In general, S is a set of explicitly functional constraints. Several specific examples of constraints are given below.

Inequality: A projectile comes from the left and hits the left side of the square.

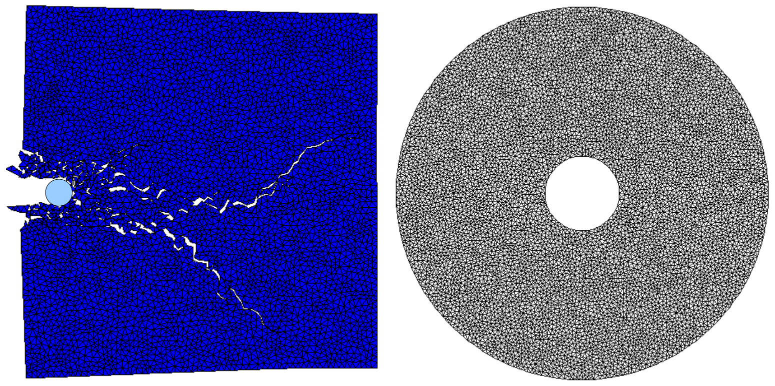


Figure 4.3: Illustration of different nonlinear constraints for the optimization

The constraints requires that no mesh node can be inside of the projectile which is represented by a circle $(x-x_i)^2+(y-y_i)^2=r^2$. So, for any mesh node (x,y), it should locate outside of the circle: $(x-x_i)^2+(y-y_i)^2 \leq r^2$. And the right side of the square cannot move to the right direction $x \leq 1$, otherwise the projectile will not fracture with the square but instead push the whole square move.

Equality: A projectile passes through the center of a high elastic material, the inside boundary can be constrained to be on the circle $x^2+y^2=r_i^2$.

The penalty method [57] replaces the constrained optimization with an unconstrained optimization of the form

$$F(\vec{x}) + mP(\vec{x}) \tag{4.3}$$

where m is a positive constant and P is a function on R^n and satisfying (1) $P(\mathbf{x})$ is continuous, (2) $P(\mathbf{x}) \geq \mathbf{0}$ for all $x \in R^n$ and (3) $P(\mathbf{x}) = \mathbf{0}$ if and only if $\mathbf{x} \in \mathbf{S}$
When S is in the form of $h(x)=0$, a commonly used P is $\frac{1}{2}(h(x))^2$, and when S is in the form of $g(x) \leq 0$, the corresponding P could be $\frac{1}{2}(\max(g(x),0))^2$

Based on this, a penalty method for the nonlinear equality/inequality constraints can be constructed,

$$F(\vec{x}) + m(\max(g(\vec{x}) - c_1, 0))^2 + m(h(\vec{x}) - c_2)^2 \tag{4.4}$$

where m is a positive constant but increases at each iteration step from zero to a number large enough until the accuracy desired is reached.

For unconstrained nonlinear optimization $Minimize f(x)$, TAO has Nelder-Mead Method, Limited-Memory, Variable-Metric Method, Nonlinear Conjugate Gradient Method, Newton Line Search/Trust Region Method.

The Nelder-Mead Method is a direct search method for finding the local minimum of a unconstrained function. It does not need gradient or Hessian information to perform the optimization, so it is easier to implement and is suitable for solving the problem whose gradient or Hessian information does not exist. The downside is, due to the lack of detail information about the function, it can be slow to converge or can even stagnate, and it does not perform well for large numbers of variables.

The limited-memory, variable-metric method calculates a positive definite approximation to the Hessian matrix from a limited number of previous iterates and gradients calculations. With this Hessian evaluation, the direction can be calculated by which a More-Thuente line search is applied to compute a step length. Then the iterate and Hessian are updated, and the process is repeated until it converges to the desired accuracy. For the best efficiency, function and gradient evaluation should be performed simultaneously when using this algorithm. This algorithm is the default unconstrained minimization solver.

The nonlinear conjugate gradient method is a generalization of the conjugate gradient method for solving symmetric positive definite linear systems of equations. The algorithm need only function, gradient evaluation and a line search. The line search method used by TAO to obtain the step length is a More-Thuente line search. For the best efficiency, function and gradient evaluation should be performed simultaneously when using this algorithm. The method incorporates automatic restarts when successive gradients are not sufficiently orthogonal.

The Newton line search method obtain a search direction from the Hessian matrix and gradient. After that a step length is calculated by More-Thuente line search and the process is repeated until the desired accuracy is reached. For the best efficiency, function and gradient evaluations should be performed simultaneously when using this algorithm.

The Newton trust-region method firstly tentatively set a search radius, which is called trust-region radius, and then obtain a best direction by a constrained quadratic programming. If the obtained direction can sufficiently reduce the value of the objective function, the step is accepted and the minimum is updated. If not, the trust-region radius is reduced, and the quadratic programming is re-solved by the updated trust-region radius. For the best efficiency, function and gradient evaluations should be performed separately when using this algorithm.

Most of our simulations are performed by nonlinear conjugate gradient and limited memory, variable-metric methods.

4.3 Mass-Conservative Crack Propagation Algorithm

As we mentioned before, the material is represented by the unstructured mesh. The mesh modeling method used by Beale and Meakin is to remove the overstressed bonds. This can create reasonable results if the stretching force applied to the material is not very big and not many cracks are created. But in our case the impact between the projectile and material modeled might be explosively huge, if the same method is used, a large amount of materials might be removed and cause the simulation results have big void region, as shown in figure 4.4.

There is one more restriction in the crack propagation scheme in Meakin's method: the new cracks can only be created to the bonds that are adjacent to the current crack surface. So we proposed and developed a mass conservative mesh manipulation method. In order to make the material conservative, when a bond is overstressed, it cannot be treated being simply deleted. One end of the bond will be split following some rules described as below. Before introducing the splitting mechanism, let's give some definitions first. When a point is on the boundary, either the original boundary or the afterward generated crack boundary, this point is called to be on $crack\ surface$. If a triangle have no other triangles connected to it, it is called an *isolated* triangle. the number of bond connected to a node is called $number\ of\ neighbour\ bonds$ of this node. the number of triangles connected to a node is called $number\ of\ neighbour\ elements$ of this node. If a bond is overstressed, and it is not a edge of a isolated triangle, this bond is selected to be the bond that one of the two ends will be split.

1. If neither of the two ends is on crack surface, the splitting end can be selected arbitrarily, here we always select the first end of the bond.

2. If one end is on crack surface and the other isn't, the splitting end is the end that is on the crack surface.

3. If both ends are on the crack surface

   (a) and both have the number of neighbor bonds greater than 2, the splitting end is selected as (1);

   (b) and one has the number of neighbor bonds greater than 2, the other has

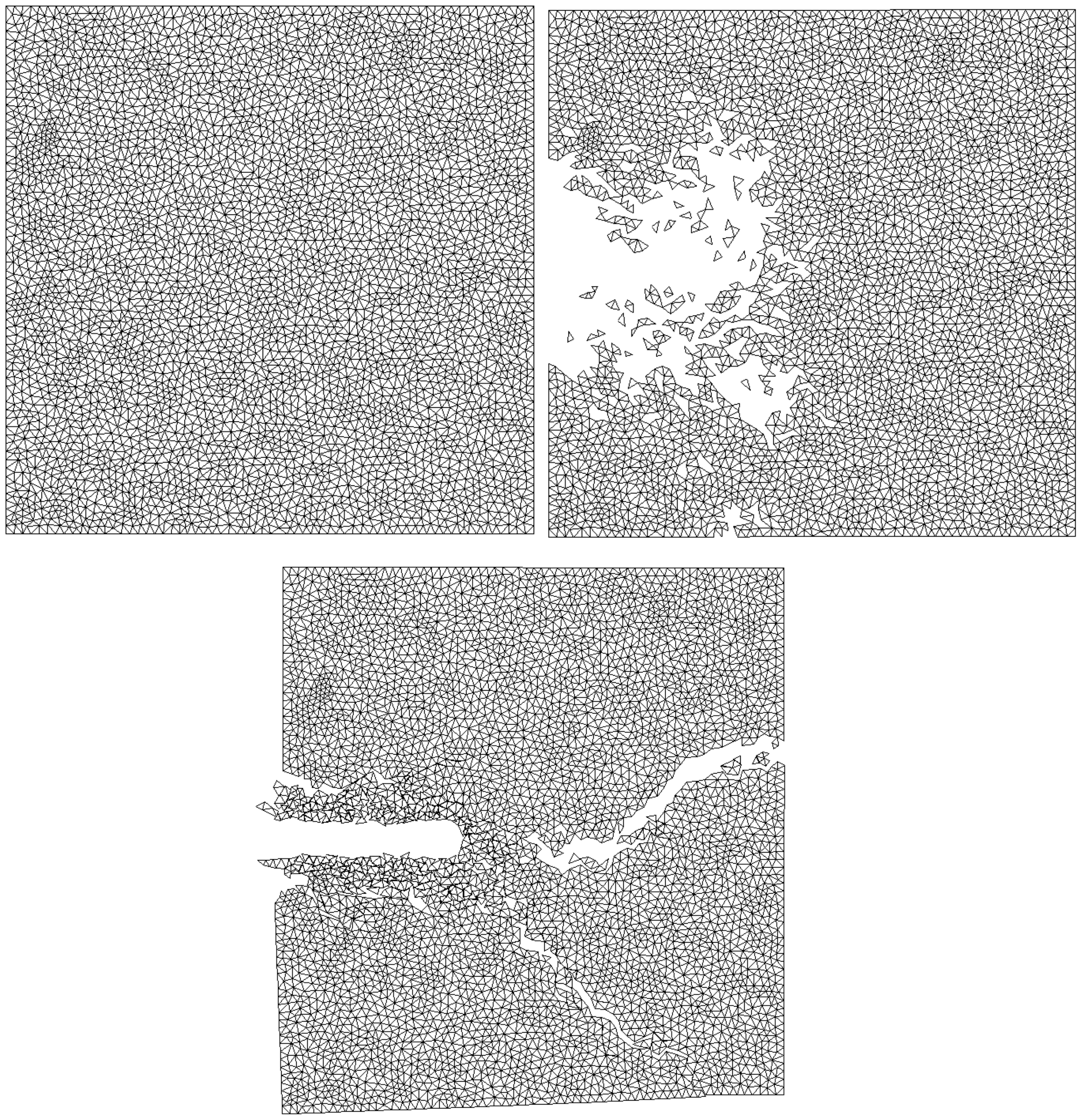


Figure 4.4: Comparison of mass-non-conservative method and mass-conservative method. Top left: initial mesh; Top right: mass-non-conservative method created large mass loss; Bottom: The mass conservative method can show the cracks under big tension.

number of neighbor bonds equal to 2, the first is selected to be the splitting end.

(c) it is impossible that both ends have number of neighbor bonds equal to 2, because in this case, the triangle that contains the bond will be an isolated triangle, or the triangle is connected to another triangle only by one vertex, which will be eliminated in the node splitting process.

After bond splitting, if two chunks are only connected by only one node, the node needs to be split, as indicated in Figure 4.5.

4.4 Folding and overlapping elimination

When the deformation is greater than the mesh size, without special treatment, the mesh may fold to achieve the minimum energy of the system, which is unphysical for brittle material, as shown in figure 4.6. During the crack propagation, there might be mesh overlapping around the crack surface, particularly when there are separated fragments in the system, and the locations of the fragments will not affect the total energy of the system. This should be eliminated, because the interaction between the cracked material is one of the reasons causing the crack propagation. In other words, whether the folding and overlapping is eliminated will affect how the crack propagates in the material. Some scientist (e.g. M. Ortiz [59]) proposed successful algorithms to solve this from the point view of force. But here we propose a method by means of energy.

Before the mesh overlap is resolved, it should be detected first. A standard way is to do the polygon(here the polygons are all triangles) collision detection pair by pair, or do the detection to all mesh elements that have at least one edge on crack

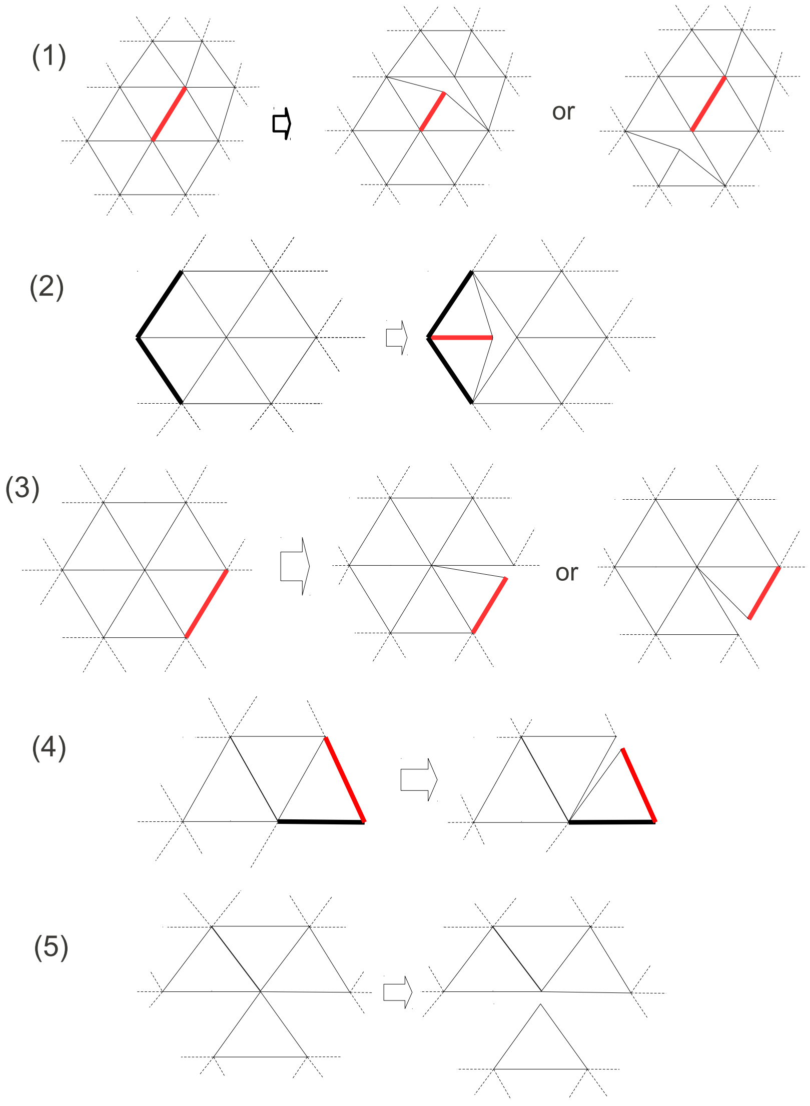


Figure 4.5: Illustration of mesh splittings in the mass-conservative fracture propagation.

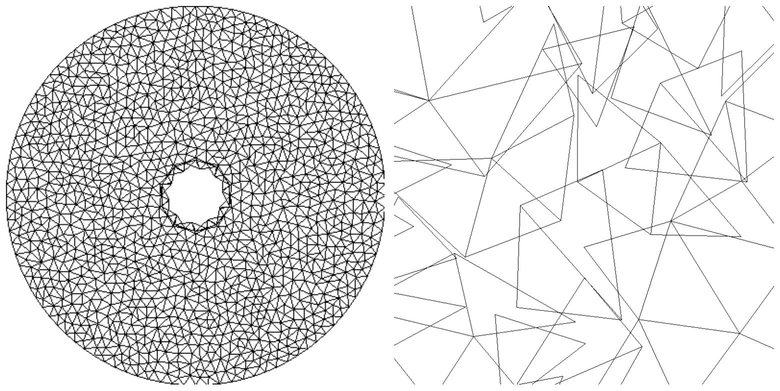


Figure 4.6: Left: Mesh overlap sample with fracture; Right: Crack overlap after fracture

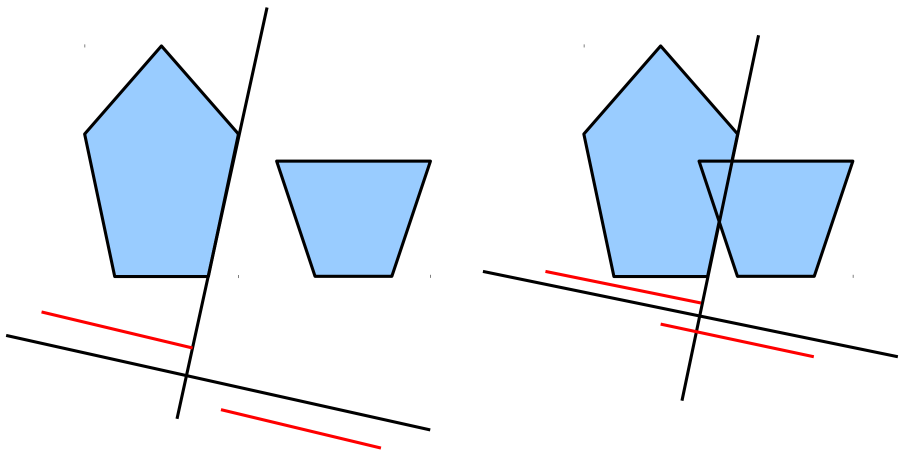


Figure 4.7: Illustration of polygon collision detection

surface. The collision detection of a pair of polygons is given by the following. Select any polygon, and draw a straight line along one edge of the polygon. Project both polygons along this straight line to a line that is perpendicular to this line, and check if those two projections intersect, as shown by red lines in the figure 4.7. Then go to next edge of the selected polygon and check by the same procedure. After all the edges have been checked, if any pair of projections of the two polygons intersects, the two polygons intersect. Otherwise, the two don't intersect.

The exact numerical implementation of the polygon collision detection leads to a very high computational cost. In our work, we use an approximate method to do the polygon collision detection and response by penalty method. First, from the coordinates of each triangle in the initial mesh, find the corresponding inradius r , and keep it to be a constant in the computation,

$$r = \sqrt{\frac{(s-a)(s-b)(s-c)}{s}} \tag{4.5}$$

where $s = \frac{1}{2}(a+b+c)$, and a, b, c are length of three edges. We can assume the inradius doesn't change during the computation since the material is brittle.

Then we calculate the distances d_{ij} between incenters of any pair of neighbor triangles T_i and T_j during the computation, if any d_{ij} is less than the summation of the corresponding inradii r_{i_0} and r_{j_0} of T_i and T_j:

$$d_{ij} < r_{i_0} + r_{j_0} \tag{4.6}$$

take it to be a penalty term of the energy function:

$$E = E + \frac{1}{2}c(d_{ij} - r_{i_0} - r_{j_0})^2 \tag{4.7}$$

Under this treatment, like in Figure 4.8, the mesh may still have overlapping, since these forces can only guarantee no overlapping among incircles. However, this can be improved by put more layers of circles in the blank area of the triangles like in Figure 4.9. If we keep doing this, the whole area of each triangle will be covered by the circles, which means this will converge to the exact solution. If we put one more layer of inscribed circles, which is three, into any triangle T_i, we can calculate the coordinates of three centers O_1, O_2, O_3 and radii r_{i_k}, where $k \in \{1, 2, 3\}$, of the three circles. Here we take calculating O_1 as an example,

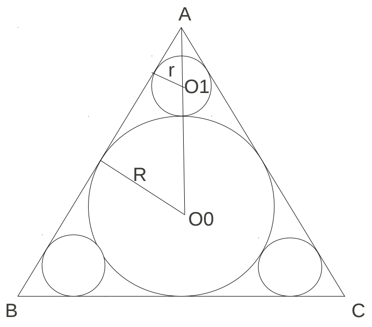


Figure 4.8: Illustration of the two-layer-inscribed-circle approximation for polygon collision detection and response.

$$\frac{|AO_0|}{|O_0O_1|} = \frac{R}{R-r} \Rightarrow \frac{|AO_0|}{R+r} = \frac{R}{R-r} \tag{4.8}$$

Since coordinates of vertex A is known, R and coordinates of O_0 are already calculated in the previous step, we can easily calculate the radius

$$r = (1 - \frac{R}{|AO_0|})\frac{R|AO_0|}{R + A|O_0|} \tag{4.9}$$

and point O_1 divide the line segment AO_0 into two parts with ratio

$$\lambda = \frac{|AO_1|}{|O_1O_0|} = \frac{r}{R-r} \tag{4.10}$$

So we have coordinates of O_1:

$$x_{O_1} = \frac{x_A + \lambda x_{O_0}}{1+\lambda}, y_{O_1} = \frac{y_A + \lambda y_{O_0}}{1+\lambda} \tag{4.11}$$

After calculating all the radii and coordinates of centers of second layer incircles, we compare $d_{i_kj_l}$, which is the distance between the center of kth circle in triangle T_i and the center of lth circle in the triangle T_j, and the summation of the two corresponding inradii $r_{i_k} + r_{j_l}$. If

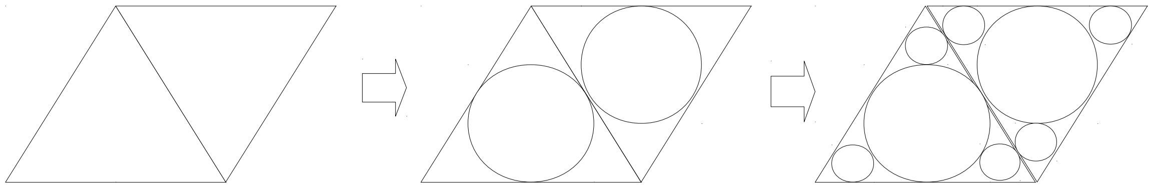


Figure 4.9: Initial triangles with one and two layers of inscribed circles.

$$d_{i_k j_l} < r_{i_k} + r_{j_l}$$

we take it to be a penalty term of the energy function like we did in the previous case. But a disadvantage of the refined method is that it is much slower. If there are N_{ele} triangles in the mesh, this will increase the number of comparison from N^2 to $(4N)^2 = 16N^2$, as well as the increase of number of penalty terms. However, the computation complexity can be reduced to Nlog(N) by using quadtree partition and search of the computation domain. If the triangles are regular triangles with side length 2, the area of the triangle is $\sqrt{3} \approx 1.732$, the area of the inscribed circle is $\pi/3$ ≈ 1.05 and the area of the inscribed circle with one more layer of inscribed circles is $\pi/3{+}3{\times}\pi/27 = 4\pi/9 \approx 1.40$. With more and more layers of inscribed circles added, this number will converge to $\sqrt{3}$.

4.5 Parallelization

4.5.1 Parallelization based on Global Arrays and parallel TAO

Global Arrays(GA) [53] is an inter-process message communication package developed by Pacific Northwestern National Laboratory. It is implemented to make the programming interfaces of the distributed-memory clusters look like shared-memory ones, but internally GA does the data communication by MPI(Message Passing Interface) and MA(Dynamic Memory Allocator). GA has been integrated into the optimization package TAO.

The well-known programming model of MPI or a typical distributed memory parallelization is organized as follows.
When a processor needs the data residing on any other processor, the first thing to do is to check the processor ID where the requested data resides, and then let the remote processor send the data to the local one. Finally, the local processor receives the piece of data from the remote processor.
At the remote processor where the needed data resides, execute:
MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm);
where **buf** is the pointer to the initial address of the sending buffer, **datatype** indicates the type of data that will be sent, **count** is the number of elements in the sending buffer, **dest** is the processor where the data will be sent to, **tag** is a integer massage tag and comm is the communicator.
And at the local processor where the data is needed, execute:
MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status);
where **buf** is the pointer to the initial address of the receiving buffer, **datatype** indicates the type of data that will be received, **count** is the number of elements in the receiving buffer, **source** is the processor where the data that will be received from, **tag** is a integer massage tag, **comm** is the communicator and status is status object.

In comparison, the programming model of GA is organized as follows.
Create a global array which holds the parallelized data. Then when a processor needs the data residing on any other processor, get the piece of data from the global array, and if the local processor updated the data it owns, put the data to the global array to keep the data on global array up-to-date.

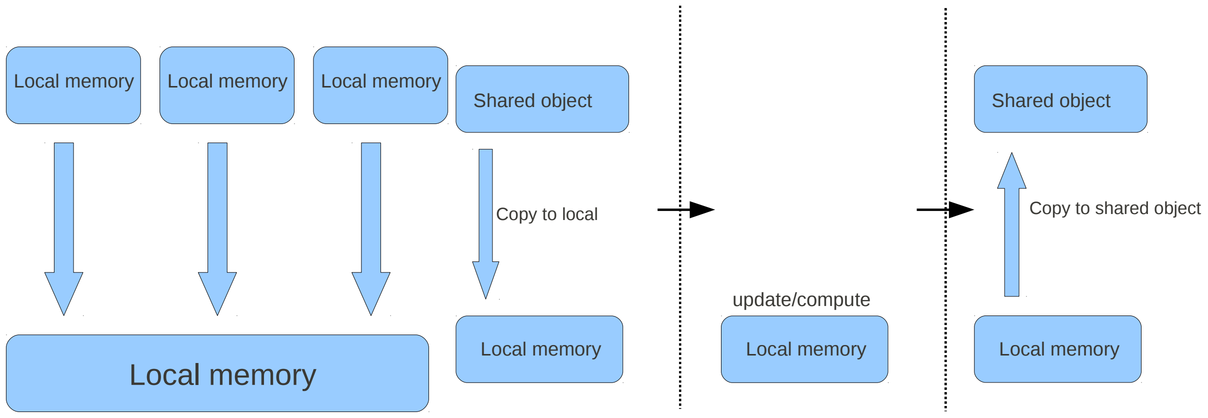


Figure 4.10: The data communication models of MPI(left) and GA(right).

NGA_Get(int g_a, int lo[], int hi[], void* buf, int ld[]);

where **g_a** is the global array handle, **ndim** is the number of dimensions of the global array, **lo/hi** is the array of starting/ending **indices** for global array section, **buf** is the pointer to the local buffer array where the data goes, **ld** is an array specifying leading dimensions/strides/extents for buffer array.

The computational domain is decomposed by fixed non-overlap domain decomposition according to the coordinates of each mesh node. As the location of cracks during the solid fracture process is unknown at the time of domain decomposition, the load balance might not be optimal all the time during the simulation. A domain with more cracks will have more computational loads than a domain with less cracks.

4.5.2 Schwartz-type Domain Decomposition for Parallelization

Usually, the domain is decomposed completely without any overlap, and message passing is required to explicitly transmit data between processors if the memory is distributed. But when the problem size becomes larger, data communication may contribute a big portion of the computational time. To overcome this, we derived a Schwartz-type alternating domain decomposition method(DDM) similar to the elliptic

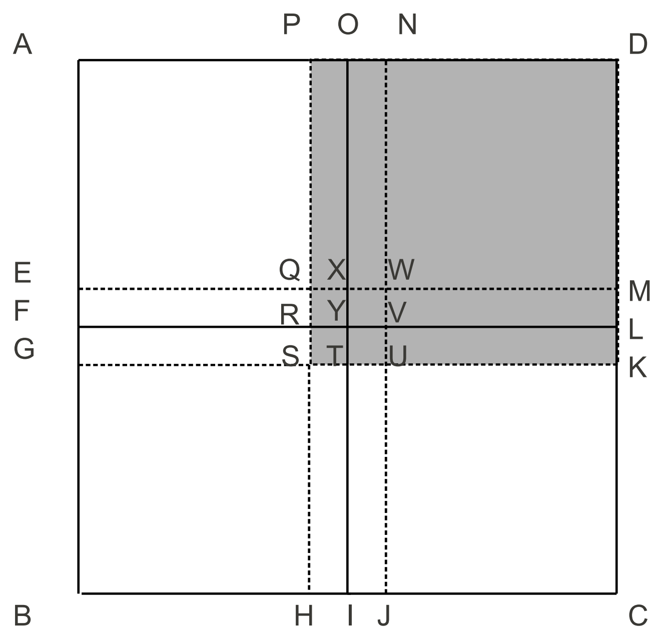


Figure 4.11: An overlapping domain decomposition with four subdomains.

boundary value problem. As shown in figure 4.11, a computational domain ABCD, if it is decomposed into four parts, we have four overlapped sub-domains: $AGUN$, $EBJW$, $HCMQ$, $KDPS$, rather than $AFYO$, $FBIY$, $ICLY$, $LDOY$ as usual. The shaded area $KDPS$ in the picture is one of them. Initially, the computation is performed on the four sub-domains independently. When it is done, the boundaries of each sub-domain are updated by its neighbor domains, and the computation continues as in the previous step until the values at the boundaries are the same to that in the neighbor domains under some small tolerance ϵ.

In this DDM, no data communication is required at each specific iteration, communication is only required between iterations. The method is easier to implement, has less data communication, but the convergence rate maybe slow.

Here, a two-subdomain test case has been analyzed, as shown in figure 4.12.

This domain is a two dimensional disk with a small hole in the center. Increasing the diameter of the inside circle, while leave the outer boundary free, the mesh nodes

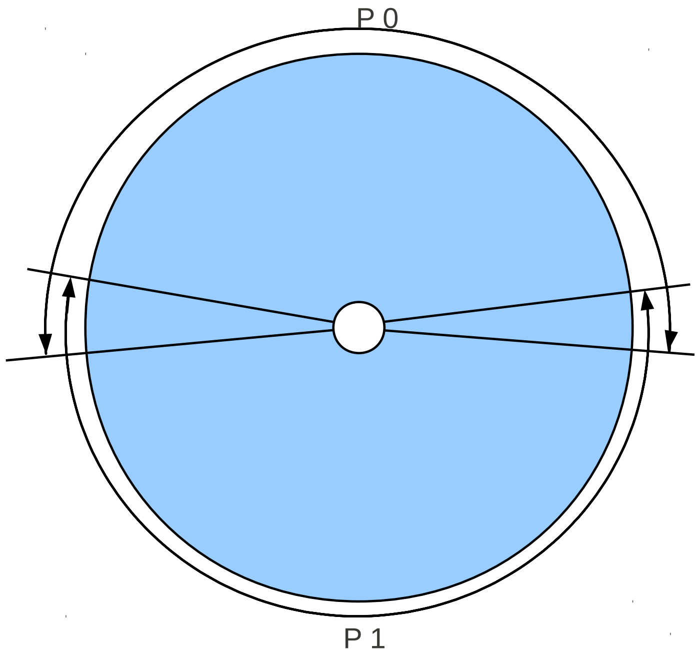


Figure 4.12: The analyzed overlapping domain decomposition with two subdomain

in the body will be redistributed to achieve a minimum energy state. The Schwartz DD is adopted in the computation of the energy minimization.
Initial values are assigned to be the locations before the boundary is moved. After each iteration, the iteration number and the reduced total energy from every processor are recorded. The result is convergent but oscillatory, and quite slow. The red curve in figure 4.13 showed the convergence history. It took almost 250 iterations to converge to the tolerance of 1.0e-6.

To achieve advantage in the computational time, the method must converge in 10 iterations. To improve the convergence, several sequence acceleration methods are researched and analyzed.

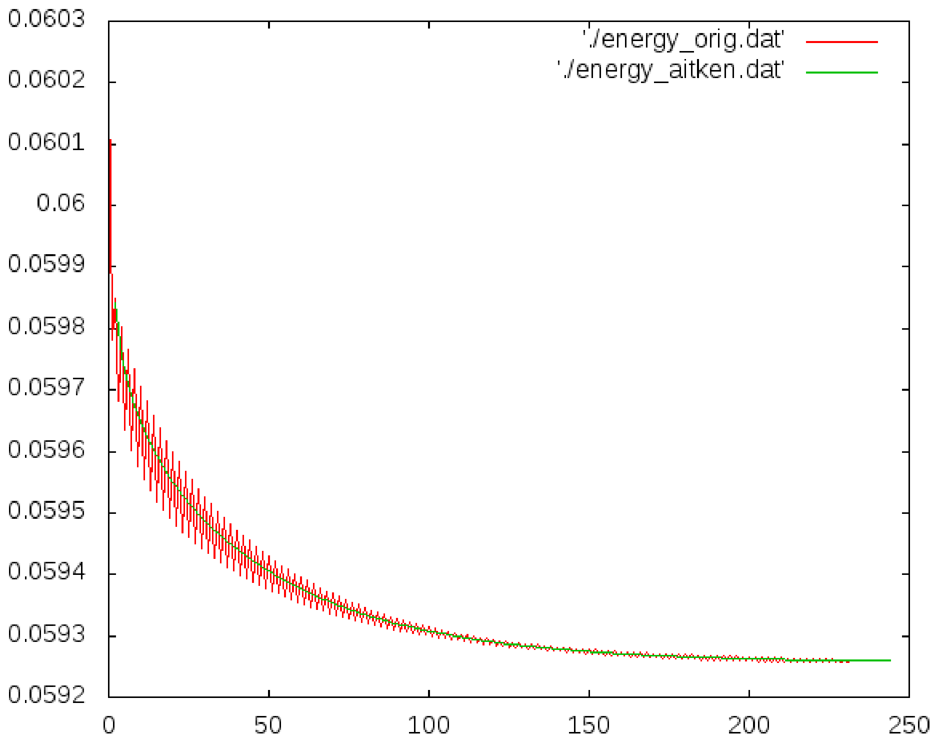


Figure 4.13: Total energy of the two subdomains v.s. the number of iterations.

4.5.3 Sequence Accelerations

Sequence acceleration techniques are critical in the improvement of rate of convergence in iterative methods. It can be categorized as linear and nonlinear, and both have the form

$$x_n^{'} = f(x_0, x_1, ..., x_{Ln}) \tag{4.12}$$

where L_n is a positive integer, $\{x_n\}$ is the original sequence and $\{x_n^{'}\}$ is the transformed sequence.

In the linear acceleration, we have

$$x_n^{'} = \sum_{i=0}^{L_n} \theta_i A_i \tag{4.13}$$

where $\sum_{i=0}^{L_n} \theta_i = 1$, and θ_i is independent of A_i.
In the nonlinear case θ_i is dependent of A_i.

In most cases, the convergence acceleration methods are applied to series that analytical forms are already known. However, in the study of our Schwartz-type DD, the analytical form is unknown, which causes difficulty in acceleration. In order to find appropriate acceleration algorithms for the sequence we have, the rate of convergence should be evaluated.

Assume a sequence $\{x_n\}$ is convergent to a value L, and the following is satisfied:

$$\lim_{k \to \infty} \frac{|x_{k+1} - L|}{|x_k - L|} = \mu \tag{4.14}$$

if $\mu \in (0, 1)$, the sequence is defined to be linearly convergent;
if $\mu = 0$, the sequence is defined to be superlinearly convergent;
if $\mu = 1$, the sequence is defined to be sublinearly convergent.

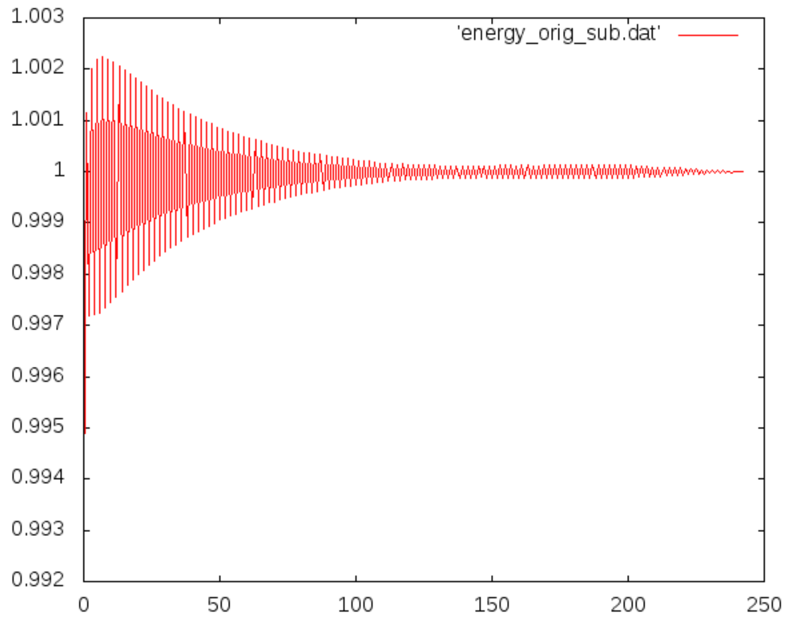


Figure 4.14: Our test shows that the sequence is sublinearly convergent.

Furthermore, if $\{x_n\}$ is sublinearly convergent and

$$\lim_{k\to\infty} \frac{|x_{k+2} - x_{k+1}|}{|x_{k+1} - x_k|} = 1 \tag{4.15}$$

holds, the sequence is defined to be logarithmically convergent.

First, we use the original data and print sequence $\left\{\frac{|x_{k+1}-L|}{|x_k-L|}\right\}$, where $k = 0, 1, 2, ..., n-1$, as shown in figure 4.14.

From the picture, we can easily know that the limit of the sequence $\left\{\frac{|x_{k+1}-L|}{|x_k-L|}\right\}$ is 1, from which we can draw the conclusion that sequence $\{x_k\}$ is sublinearly convergent.

Furthermore, the following sequence has been printed:

$$\left\{\frac{|x_{k+2} - x_{k+1}|}{|x_{k+1} - x_k|}\right\}$$

as in figure 4.15.

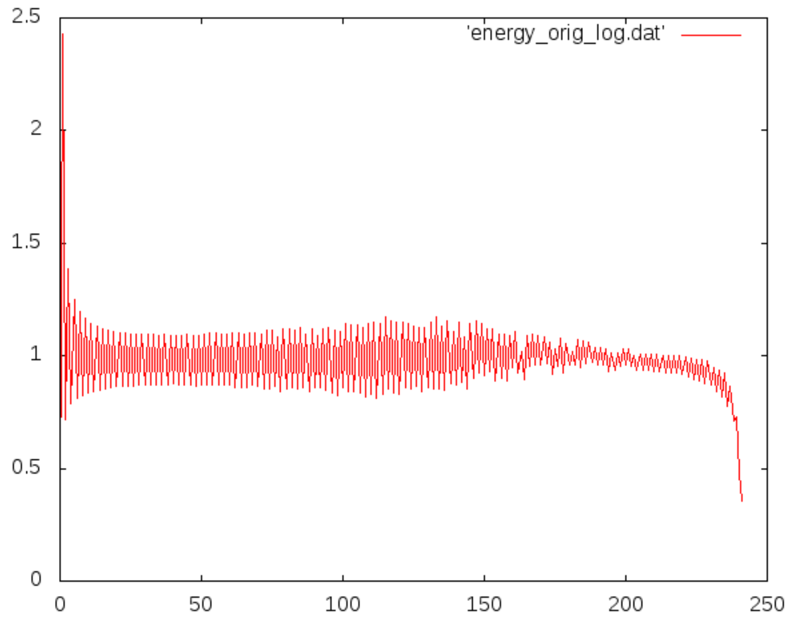


Figure 4.15: Further more, our test shows that the sequence is almost logarithmically convergent.

The values are around 1 except a few on the tail, which is very likely the result of the numerical precision. Therefore, we can assume that the sequence is logarithmically convergent.

The typical acceleration algorithms [41] for logarithmically convergent sequence include Wynn's ρ algorithm, Levin's μ-Transformation, Lubkin's Iterated W-Transformation and Brezinski's θ Algorithm. All those algorithms need a broad range of values in the original sequence to predict even one term. Here, as an example of them, we give more details of Wynn's ρ Algorithm, which performed the best in all 11 tests taken by Valko [40]. It is given by,

$$\begin{gathered}\rho_{-1}^{(n)} = 0, \rho_0^{(n)} = x_n, n \geq 0, \\ \rho_k^{(n)} = \rho_{k-2}^{(n+1)} + \frac{k}{\rho_{k-1}^{(n+1)} - \rho_{k-1}^{(n)}}, k \geq 1\end{gathered} \tag{4.16}$$

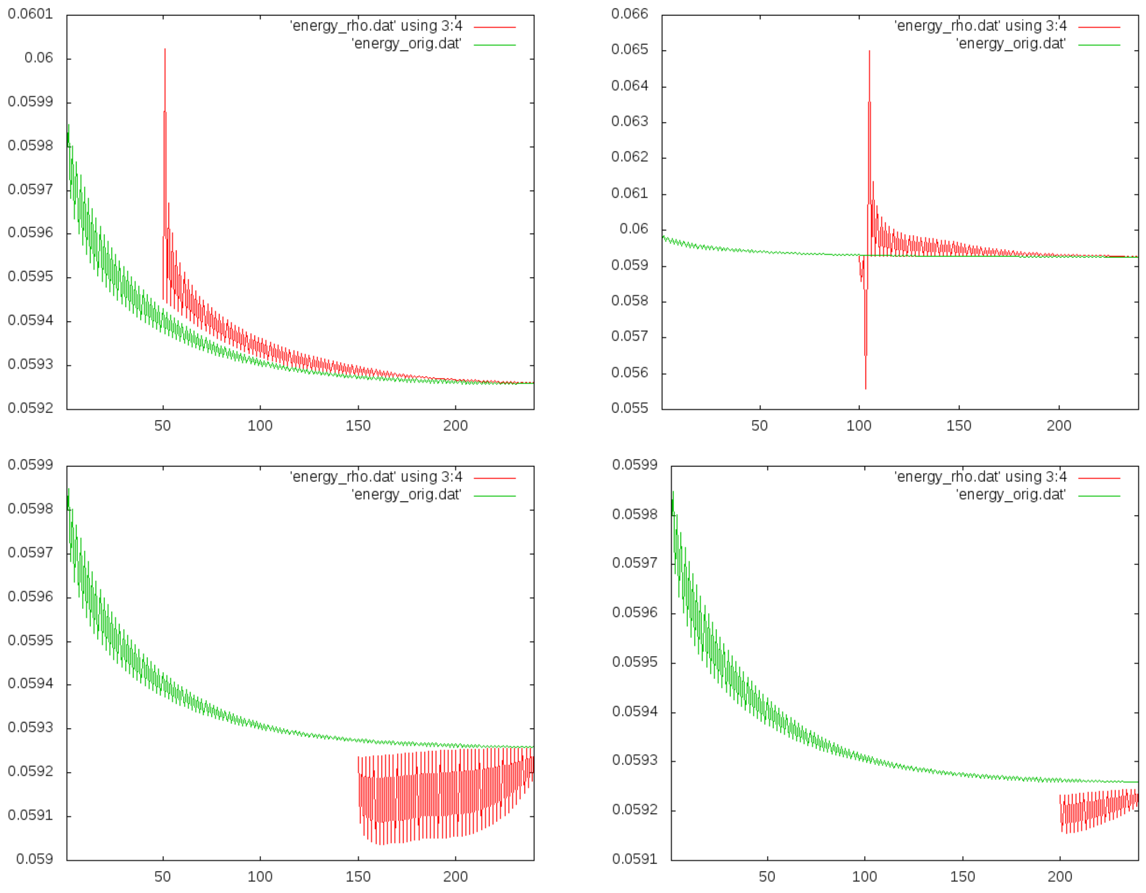


Figure 4.16: The acceleration using Wynn's ρ algorithm with different stencil length k. Top left: k = 50; Top right: k = 100; Bottom left: k = 150; Bottom right: k = 200.

It explicitly depends on only three terms of previous calculated results, but implicitly, it depends on from x_0 to x_k. Our result show that the did no improvement to the sequence we obtained. The pictures in figure 4.16 are convergence history for $k = 50, k = 100, k = 150, k = 200$ respectively,

When k is small($k \leq 100$), the accelerated sequence converged to the same value as the original one, but the speed is even worse. When k is big($k \geq 150$), the accelerated sequence even didn't converge to the same value as the original one, which might be caused by the accumulated errors in the iteration. To reduce the accumulated errors as much as possible, Valko [40] used 50 significant digits and Tuyl [41] used 28. In our tests, we have 22 significant digits, this is why the accumulated error is doubted. However, the accumulated error is not the main reason for the slow convergence rate when $k \leq 100$.

We also tried to eliminate the oscillation in the original sequence by applying Aitken Δ^2 process for only one iteration. Aitken Δ^2 process is an algorithm to do the acceleration with little effort, which uses only three consecutive terms from the original sequence(or previous sequence if iterative aitken method is been used) to construct the new sequence. Suppose there is a series $\{x_n\} \rightarrow x$ of linearly convergence. The new series $\{x_n^{'}\}$ are constructed by:

$$x_n^{'} = \frac{x_{n+2}x_n - x_{n+1}^2}{x_{n+2} - 2x_{n+1} + x_n} \tag{4.17}$$

This algorithm is for linearly convergent sequence, but can be used to successfully eliminate oscillations in our sequence, as shown by the green curve in the figure 4.13. Apply the ρ algorithm to this oscillation eliminated sequence doesn't work as well, plus the accelerated sequence has a couple of large spikes in the middle.

But what we want to have is a accelerator that works efficiently only with a small number (e.g. < 10) of terms in the previous sequence, because the optimization process with our overlapping DD is an iterative method, the values of the sequence can only be obtained sequentially one by one. If the accelerator needs too many terms in previous sequence, it becomes ineffective, since we have already spent significant computational time to obtain the accurate enough value. Delahaye et al. [42] showed that a general accelerator that can successfully work on any logarithmically convergent sequence doesn't exist. A new sequence acceleration technique need to be created for this sequence in the future.

But this method could be relatively efficient compared to the non-overlapping do-

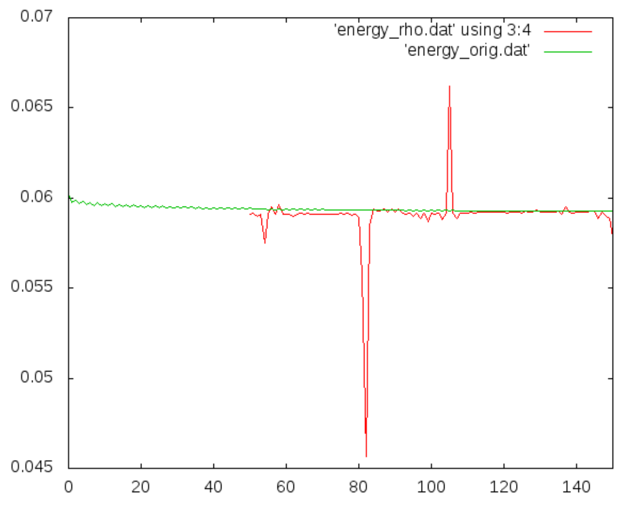


Figure 4.17: The acceleration using Wynn's ρ algorithm to the oscillation eliminated sequence.

main decomposition when there are a large number of subdomains. For example, if the sequential time complexity of the problem is N^2, with p processors, the conventional method can theoretically reduce the time complexity to N^2/p at most. Furthermore, the optimization methods usually have small parallel efficiency when the number of processors becomes large. But with the overlapping method, at each iteration the time complexity is reduced to $(N/p)^2$ exactly, because no data communication is required at each iteration and only small data communication between each two iterations. This leaves us big room for the number of iterations in the overlapping domain decomposition. The convergence improves with larger overlapping to subdomains, but at the same time this will increase the size of each subdomain. The convergence is also very fast for some special cases of extra symmetry in boundary conditions. Another domain decomposition method which uses the same idea is change the domain decomposition after each iteration, as in figure 4.18.

This method can be very efficient with shared memory machines, since no data communication is required and the computation is total local without interaction among

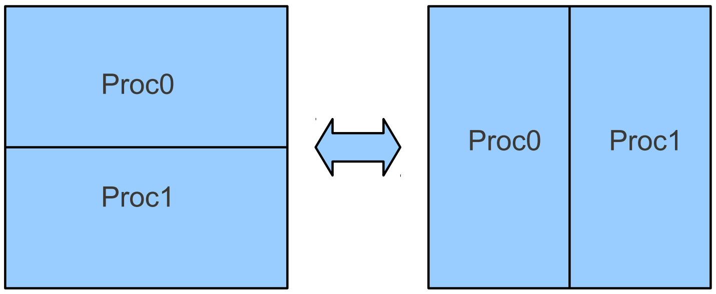


Figure 4.18: switch domain decomposition between two iterations

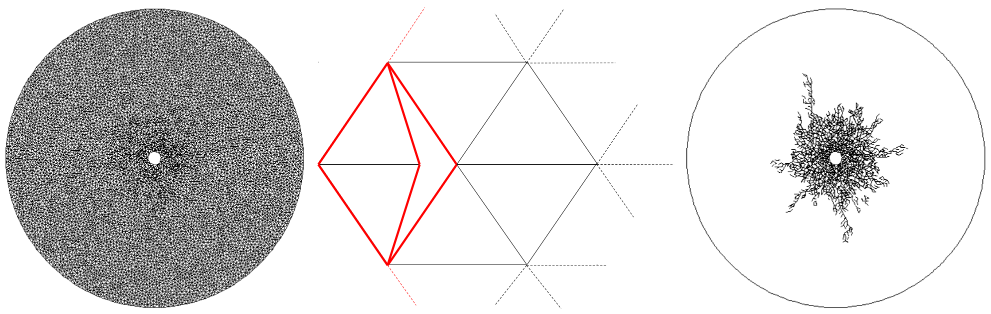


Figure 4.19: Visualization of crack surface in 2D

processors.

4.5.4 Data Analysis and Visualization

For the purpose of crack visualization in 2D, the mesh edges which are on the crack surface or initial boundary are labeled and redirected to a separate set of files. Only the crack surfaces are visualized. See figure 4.19.

In 3D, the visualization is much more complex and any one method is difficult to display enough information to the simulation results. Therefore, we developed three different types of visualizations: a particle based method, an iso-surface method and a normal plane method which are complementary to one another.

If any edge overstressed and afterward be broken, the two end vertices of this edge will be labeled to be displayed in a different color. Furthermore, in order to see the

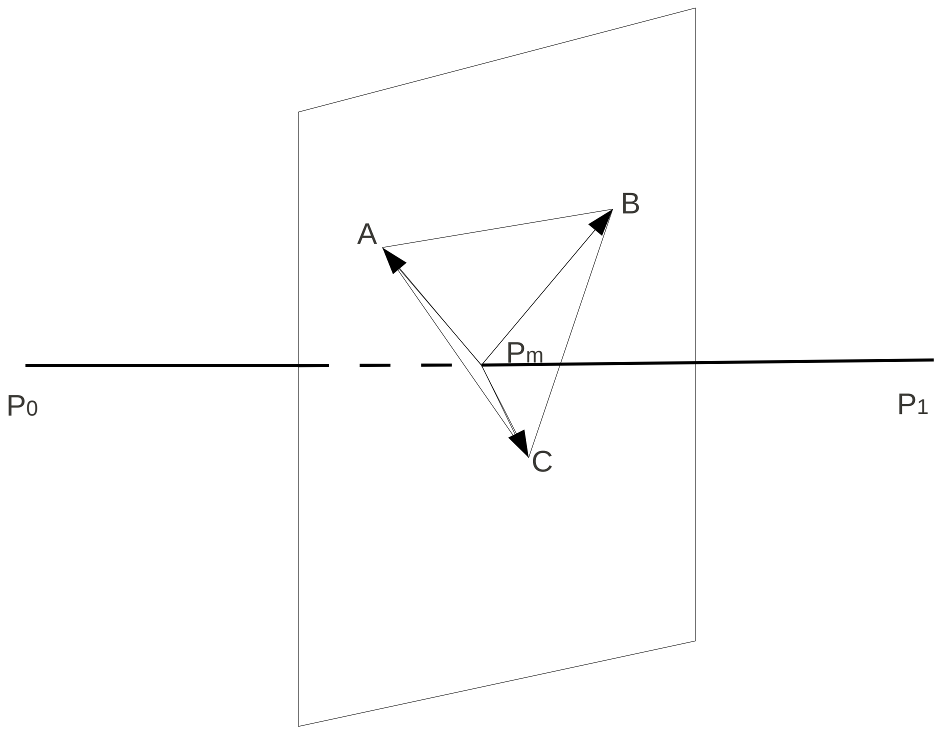


Figure 4.20: The calculation and visualization of normal plane of the broken bond.

interior cracks, the normal plane across its middle point of the edge is visualized.

The coordinates of the edge end points $P_0(x_0, y_0, z_0)$ and $P_1(x_1, y_1, z_1)$ are known, and by taking their average we will have the coordinates of the middle point $P_m(x_m, y_m, z_m)$, where $x_m = \frac{1}{2}(x_0 + x_1), y_m = \frac{1}{2}(y_0 + y_1), z_m = \frac{1}{2}(z_0 + z_1)$
also the vector $\overrightarrow{P_0P_1}$ is:

$$\overrightarrow{P_0P_1} = (x_1 - x_0, y_1 - y_0, z_1 - z_0) \tag{4.18}$$

Three unparallel vectors that are perpendicular to $\overrightarrow{P_0P_1}$ can be easily derived,
$\vec{v_0} = (0, z_0 - z_1, y_1 - y_0)$, $\vec{v_1} = (z_0 - z_1, 0, x_1 - x_0)$, $\vec{v_2} = (y_0 - y_1, x_1 - x_0, 0)$
In the case of the length of those vectors might be very large, which may result in large aspect ratio in the final triangle, we normalized all of them to a length d. The

normalized vectors are $\overrightarrow{P_mA}$, $\overrightarrow{P_mB}$, and $\overrightarrow{P_mC}$ as shown in the figure 4.20. Now, the coordinates of A, B, C can be easily calculated by

$$\begin{aligned}
A &: (x_A, y_A, z_A) = (x_m, y_m, z_m) + \overrightarrow{P_mA} \\
B &: (x_B, y_B, z_B) = (x_m, y_m, z_m) + \overrightarrow{P_mB} \\
C &: (x_C, y_C, z_C) = (x_m, y_m, z_m) + \overrightarrow{P_mC} \qquad (4.19)
\end{aligned}$$

Triangle ABC will be drawn to represent the normal plane, which is used to visualize the crack surface, of the broken edge.

In computational fluid dynamics simulation, iso-surface visualization technique is commonly used. The fluid density at each mesh node is calculated and then use the interpolation to form the surfaces that have the same fluid density. Because the mesh-free particle method is used here, a mesh needs to be created to do the interpolation. An intuitive and less difficult way is to set each particle density to 1, and map the particle density by tri-linear interpolation to the corners of a structured mesh cell.

Assume cubic shown in figure 4.22 is a 3D mesh cell with values known at the eight mesh nodes. The trilinear interpolation is to find the value at a point P who is in the interior of the cubic according to those eight values based upon the assumption that the values change linearly in the 3D space. The trilinear mapping talked here is a reverse process of the trilinear interpolation. First, the density value f at point P is mapped to the two ends P_0 and P_1 of line segment P_0P_1,

$$f(P_0) = f(P)\frac{(z_2 - z)}{(z_2 - z_1)}, f(P_1) = f(P)\frac{(z - z_1)}{(z_2 - z_1)} \qquad (4.20)$$

then map the value at P_0 to the ends of line segment $P_{00}P_{10}$ and value at P_1 to the

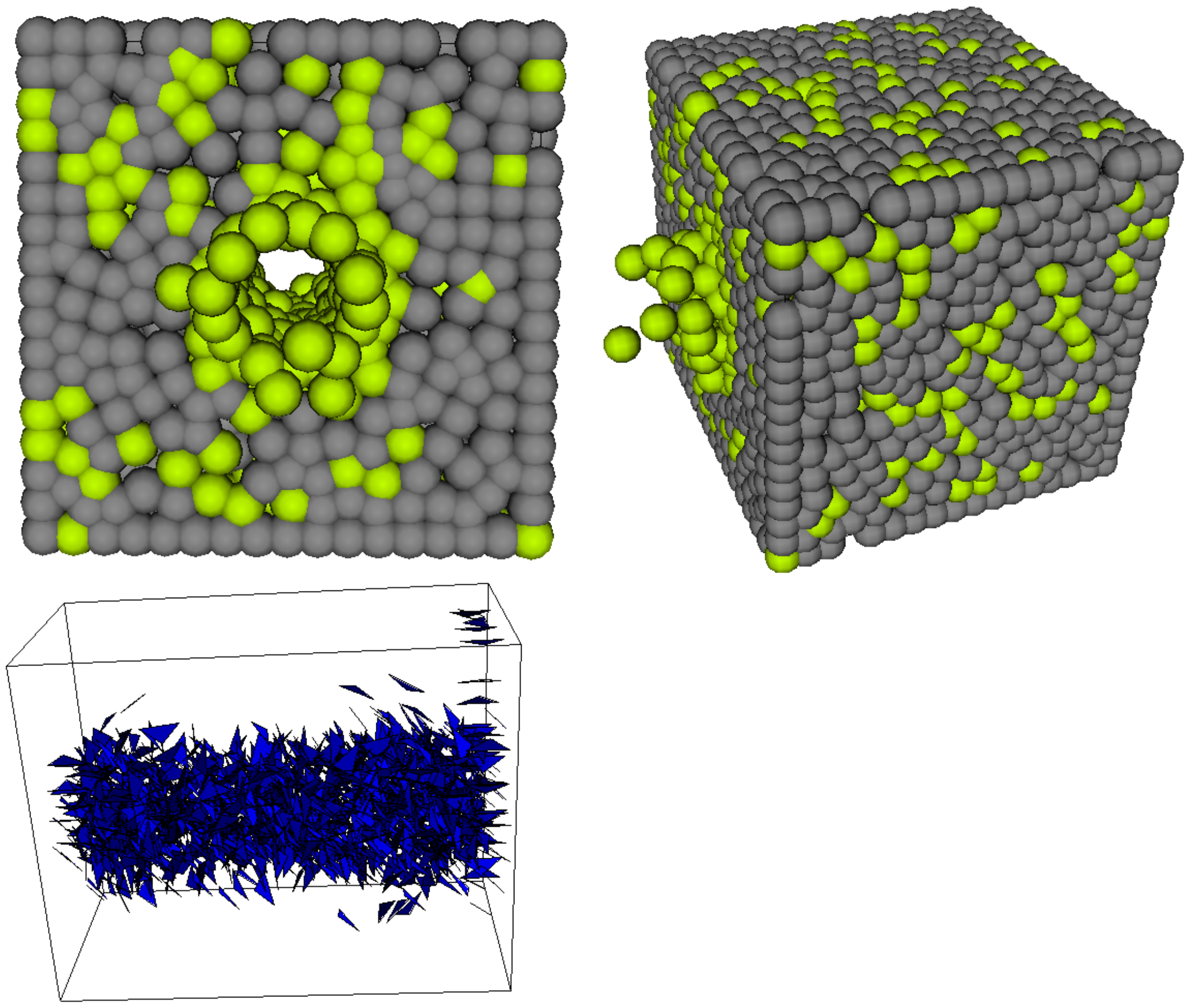


Figure 4.21: 3D visualization. Top: Two different views of a particle based crack visualization. Green particles are that have connected bonds broken. Bottom: Crack visualization by the normal planes.

ends of line segment $P_{01}P_{11}$

$$
\begin{aligned}
f(P_{00}) = f(P_0)\frac{(y_2 - y)}{(y_2 - y_1)}, f(P_{10}) = f(P_0)\frac{(y - y_1)}{(y_2 - y_1)} \\
f(P_{01}) = f(P_1)\frac{(y_2 - y)}{(y_2 - y_1)}, f(P_{11}) = f(P_1)\frac{(y - y_1)}{(y_2 - y_1)}
\end{aligned}
\tag{4.21}
$$

Finally, map values at $P_{00}, P_{10}, P_{01}, P_{11}$ to the ends of line segments $P_{000}P_{001}$, $P_{010}P_{011}$, $P_{001}P_{101}$, and $P_{011}P_{111}$, which are exactly the eight cell vertices we want to map to by the following.

$$
\begin{aligned}
f(P_{000}) = f(P_{00})\frac{x_2 - x}{x2 - x1}, f(P_{001}) = f(P_{00})\frac{x - x_1}{x2 - x1} \\
f(P_{010}) = f(P_{10})\frac{x_2 - x}{x2 - x1}, f(P_{011}) = f(P_{10})\frac{x - x_1}{x2 - x1} \\
f(P_{001}) = f(P_{01})\frac{x_2 - x}{x2 - x1}, f(P_{101}) = f(P_{01})\frac{x - x_1}{x2 - x1} \\
f(P_{011}) = f(P_{11})\frac{x_2 - x}{x2 - x1}, f(P_{111}) = f(P_{11})\frac{x - x_1}{x2 - x1}
\end{aligned}
\tag{4.22}
$$

All mappings a mesh node received are accumulated. Figure 4.23 is a visualization of a unit cubic being pulled apart. Note that the upper part of the body is removed in order to see more interior structures of the fracture.

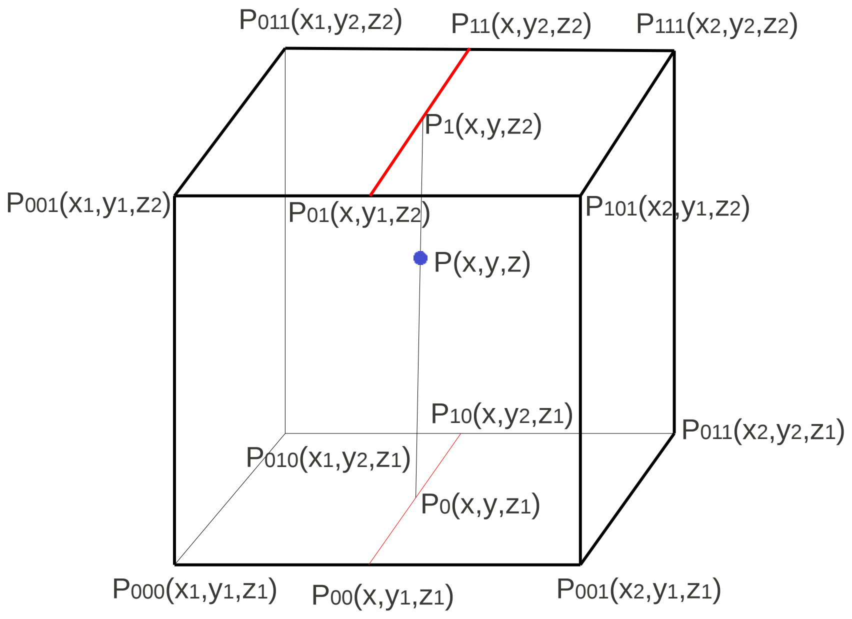


Figure 4.22: How the density of a particle is distributed to the eight mesh cell corners by trilinear interpolation.

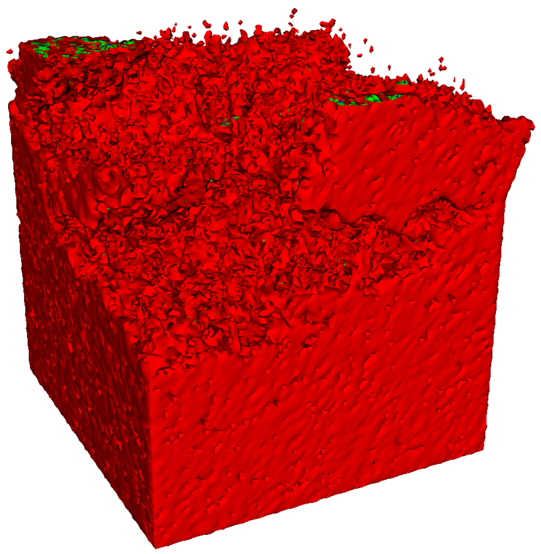


Figure 4.23: A 3D crack visualization by iso-surface method

Chapter 5

Numerical Results, Validation, and Applications

5.1 Prince Rupert's Drops

Prince Rupert's Drops are one type of glass drops that is named after Prince Rupert of Germany. The drops are of high internal stress and can be created by pouring melted glass into cold water, which cools down the surface layers very fast. The glass forms into a tadpole-shaped droplet with a long and thin tail. The outer surface of the drop was cooled very fast by the water with the solidification of the shell, while the temperature of the interior of the drop remains significantly hotter. When the interior of the glass finally cools down, it generates very big tensions among the granules of the interior, because of the solidification of the shell, and these tensions cannot be released. Rupert's drops show two perplexing mechanical properties: the main body of the drop can sustain a large extreme impact, but the tail can be quite easily broken by a small external pressure and the whole drop will disintegrate within microseconds producing a fine powder. It is widely accepted now that the surface of a Rupert's drop is in compression and the interior is in tension. From the experiments performed by Chandrasekar [58], the surface compressive layer of drops of head diameters 6.4-7.9mm was estimated to be $\sim$ 0.9mm and the fragment size were in the range of 15μm to 2.3mm. Chandrasekar [58] showed that the drop having a head with the diameter of 6-8mm could be pressed between two hardened steel rollers to a load of as large as 15,000N without breaking. If apply a very small perturbation or damage in the drop tail, the drop will disintegrate with a speed of fracture zone propagation as large as 1450-1900 m/s and reasonably constant[58]. Most of the crack propagation features seen in a disintegrating drop are very similar to that observed from tempered glass.

Here, we use our model to simulate this process in 2D. After the mesh is loaded, we set the tension of central zone artificially and apply a disturbance to the tail.

In the initial mesh, the left part is a unit circle, and the tail of the drop extends to x=6cm to the right. Then a perturbation is introduced by moving the part of mesh in the right of vertical line x=5cm a amount of 0.01cm to the left. The cracks propagate very quickly to the left side, and the whole drop become completely disintegrated in a very short time. See figure 5.1.

5.2 Fracture of Thin Discs hit by Projectiles

In this section, we report 2D simulation results of the fracture of thin brittle disks hit by high velocity projectiles. The characteristic feature of such fracture process is the bifurcation of the fracture dynamics from the growth of the comminuted zone to the propagation of isolated radial cracks. Figure 5.2 shows experimental images of a glass sheet and a SiC plate hit by projectiles. In both cases, a completely fractured (comminuted) zone is located around the projectile hole, and a number of distinct cracks extend from this comminuted zone in the radial direction. The goal of our

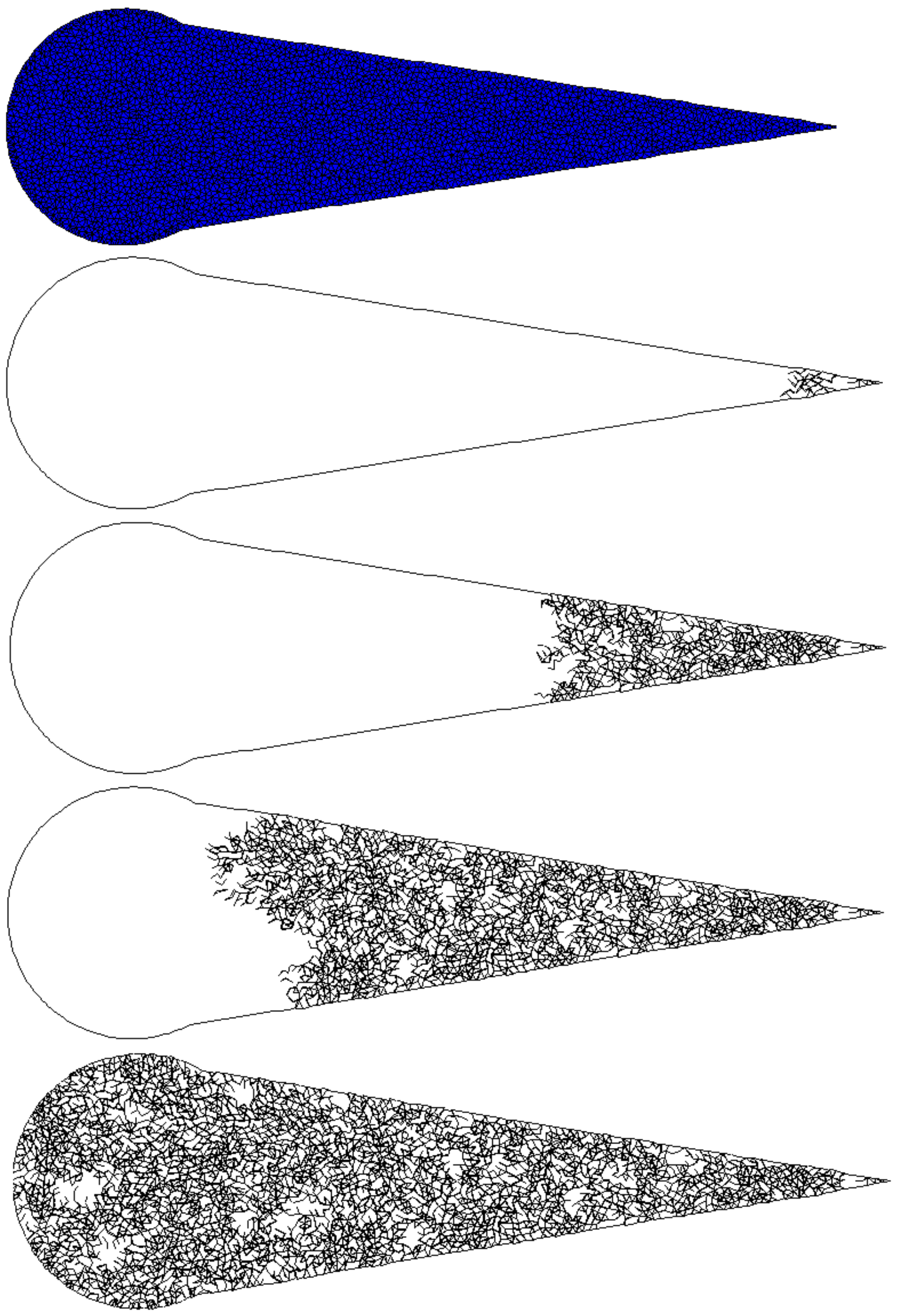


Figure 5.1: Top: The initial mesh; Second top to bottom: Simulation of the crack propagation of Prince Rupert's Drop with its tail disturbed.

simulations is to reproduce this characteristic behavior of the fracture process and investigate its dependence on material properties and other parameters.

We represent a thin brittle disk with a 2D meshed circle of 10 cm diameter. When it is hit by a projectile with the diameter of 0.1 cm, a very small seed hole is initially created in the center of the disk. We increase the radius of the hole to its final value by means of constraints applied to the inner circle boundary conditions while there are no constraints applied to the outer boundary of the disc. After each increase of the hole, the energy minimization and crack formation algorithms are applied. As it is shown in Figure 5.3, our simulations exhibit both the formation of the comminuted zone and then the growth of 7 radial cracks. The mesh containing 8,000 nodes was used in this simulations, the critical strain was set to 0.1%, and the step size of the hole increase was chosen as 0.01 cm. By changing the critical strain to 1% and keeping other parameters fixed, we obtained the results shown in Figure 5.4 (a). The increase of the critical strain corresponds to the change of the brittle properties (the material becomes less brittle), resulting in the reduction of radial cracks from 7 to 3. This qualitatively agrees with observations. In the right part of figure 5.4, the simulation result corresponding to the mesh size if 5,000 nodes is shown. It is clear that the main features of the fracture process are independent of the mesh size: the size of the comminuted zone and the number of cracks are the same as in the left image.

In the last test, we check the relationship between cracks and the projectile velocity by decreasing the step size of the hole increments from 0.01 cm to 0.001 cm and keeping the mesh size of $\sim$ 5,000 nodes and the critical strain of 1%. This decrease of the step size corresponds to the reduction of the projectile velocity and results in the formation of only two cracks.

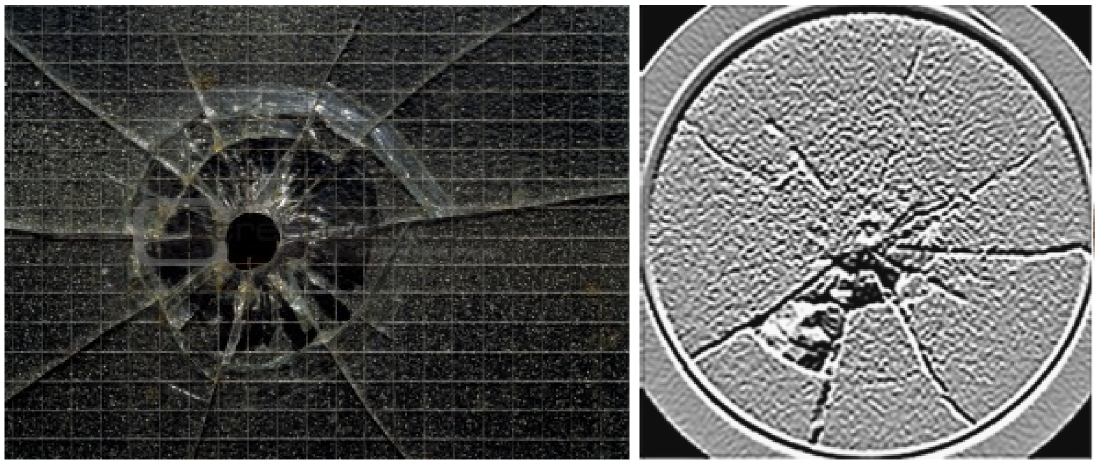


Figure 5.2: Left: A piece of glass hit by a projectile; Right: A chunk of SiC hit by a projectile [60](use with permission from the author).

5.3 2D Simulation of Thick Wall Fracture

Another set of tests is a projectile with diameter 0.01m hit a thick wall in 2D, which is represented by a 1m $\times$ 1m mesh, see figure 5.6. The boundary condition for the simulation shown in the left picture is the top and bottom is only allowed to move horizontally. The boundary condition for the simulation in the right picture is the right boundary of the square cannot move to the right direction. The critical strain is 0.003. The projectile comes from left and passes through the center of the square. The result obtained by utilizing the one-layer inscribed circle approximation algorithm is shown in the left picture, and the result obtained by utilizing two layers of inscribed circle approximation is shown in right picture. Apparently, the picture simulated with two layers of inscribed circles is more brittle.

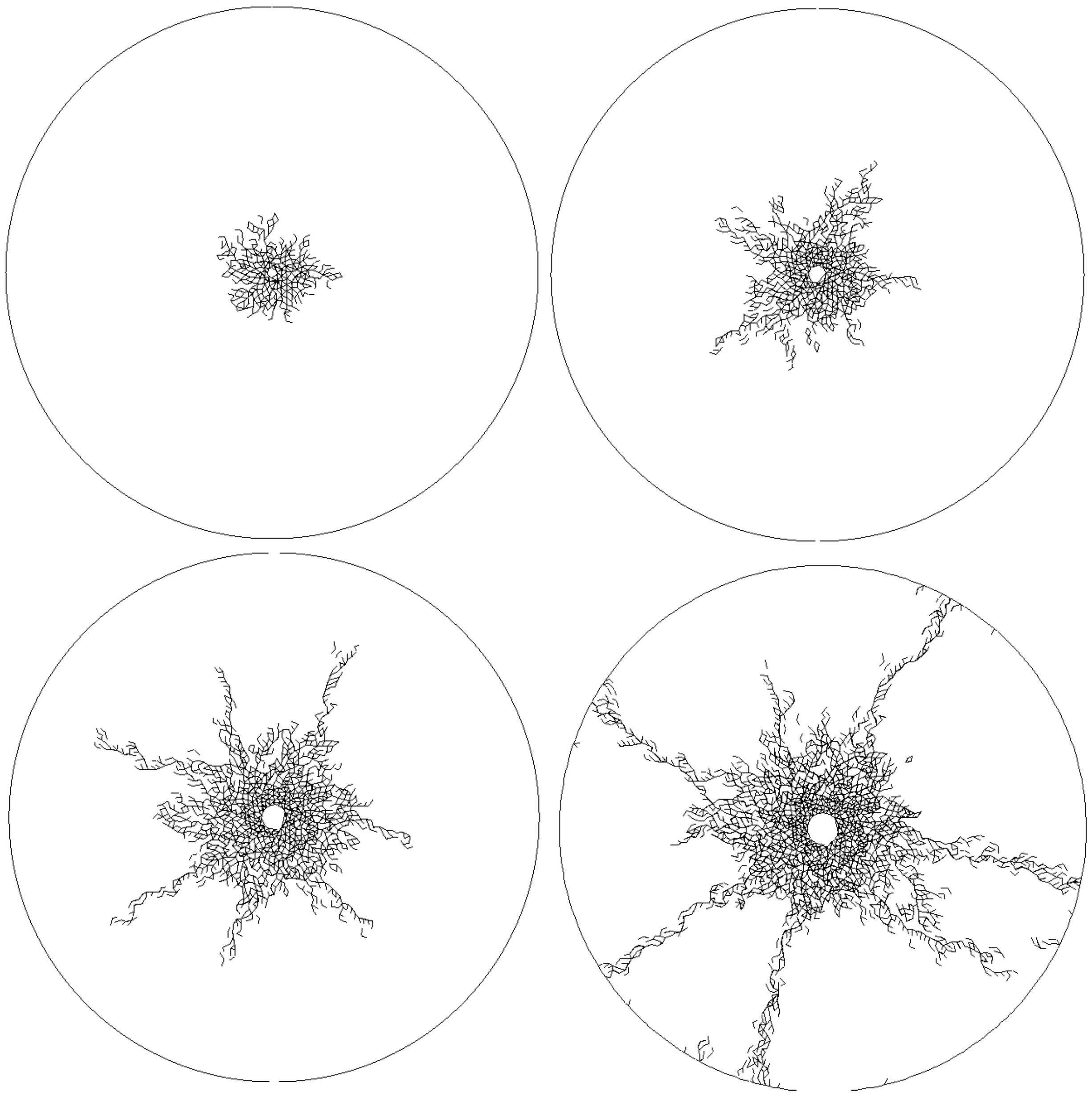


Figure 5.3: Simulation of crack formation and propagation in a 2D unit disc with critical length 0.1%, mesh size $\sim$ 8,000 and step size 0.001m.

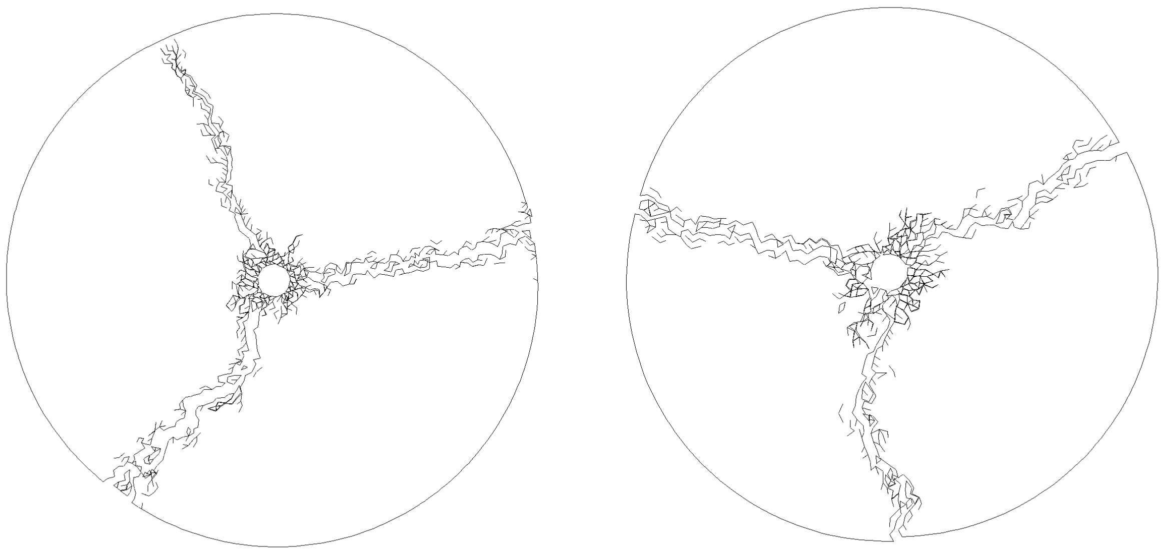


Figure 5.4: Simulation of crack formation and propagation in a 2D unit disc with critical length 1%, mesh size $\sim$ 8,000 and step size 0.001.

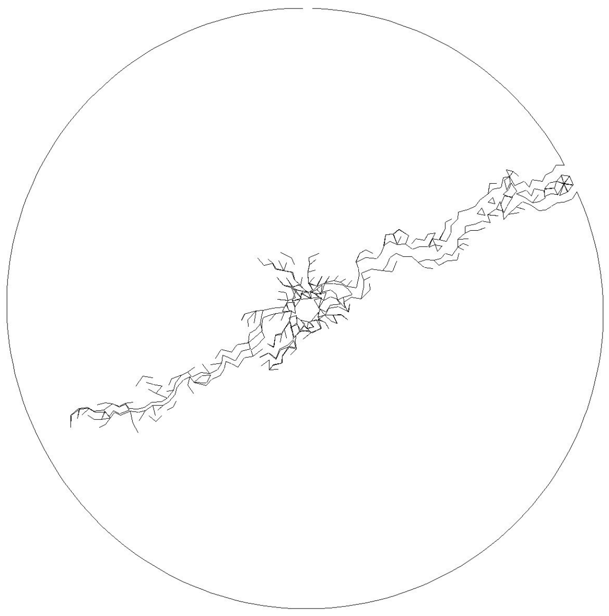


Figure 5.5: Simulation of crack formation and propagation in a 2D unit disc with critical length 1%, mesh size $\sim$ 5,000 and step size 1e-4cm

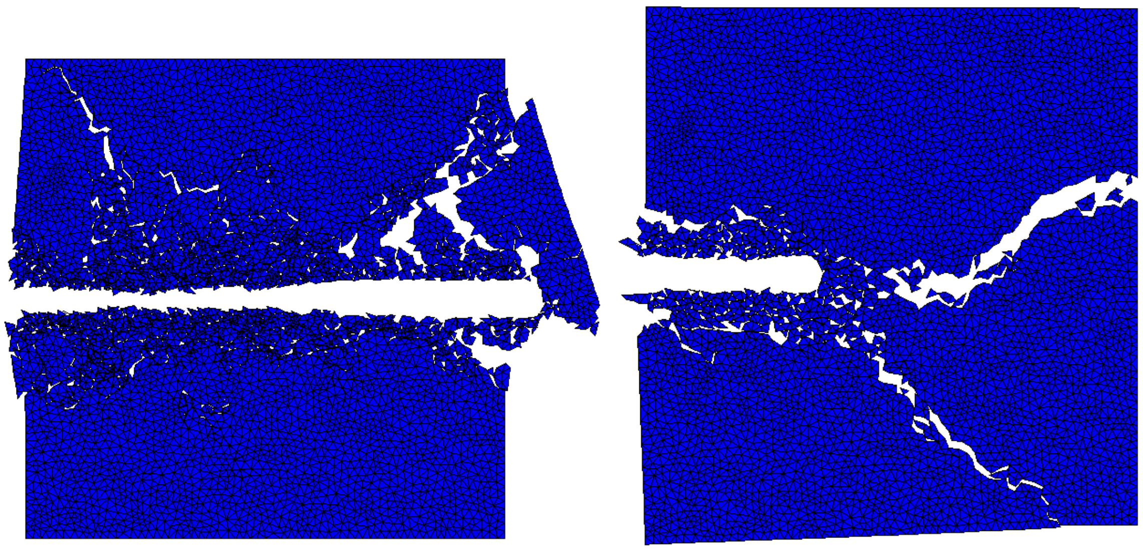


Figure 5.6: 2D simulation of a projectile hit a thick wall. Left: result obtained by one-layer inscribed circle; Right: result obtained by two-layer inscribed circle

5.4 3D Simulations

In our 3D simulation, we use a particle mesh method. A tetrahedral mesh is generated and the material volume is stored in the mesh nodes. In our first 3D simulation a unit cube is stretched until the fracture occurs. The mesh has approximately 320,000 mesh nodes, and run by 32 processors.

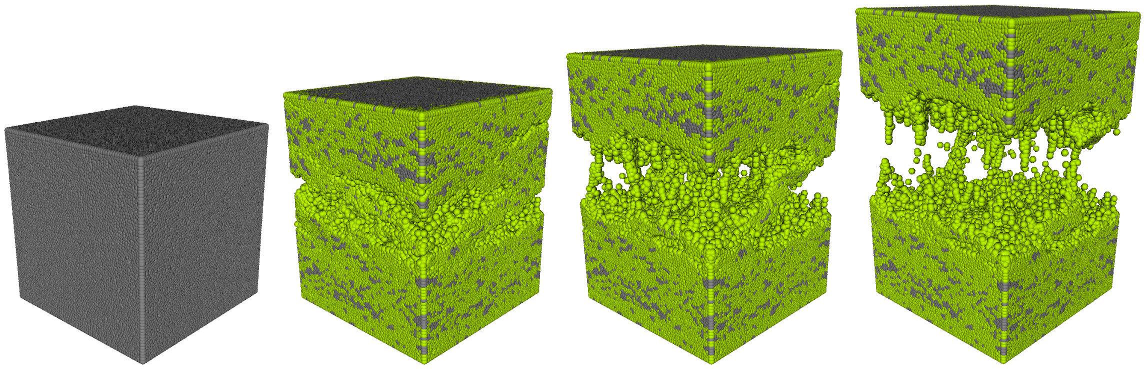


Figure 5.7: 3D cube pulled apart visualized by particles

To keep the total volume conserved in the simulation, a method to prevent particle

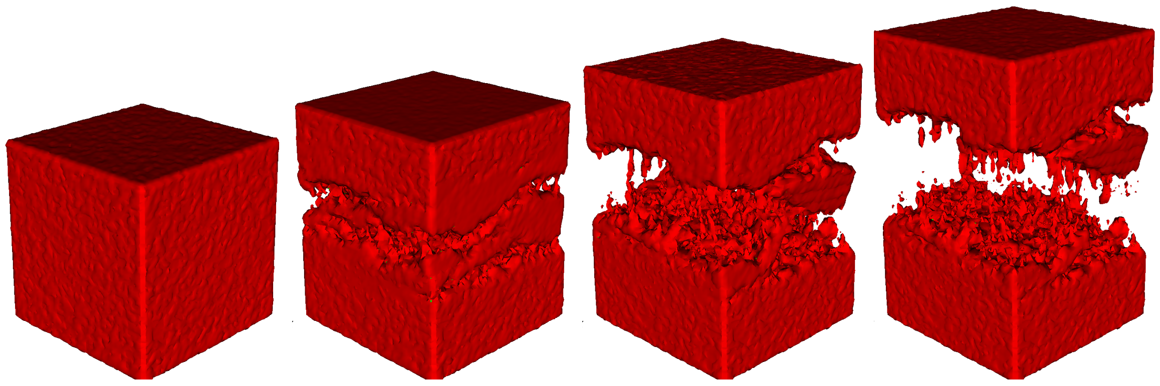


Figure 5.8: 3D cube pulled apart visualized by iso-surface

collapse is developed. The accurate way is to do polyhedron collision detection and response which is very computationally intensive. To speed up computation, we developed an approximate method. For each node, the average length of all edges that the node connected to is calculated.

$$L_{avg} = \sum_{i=1}^{i=N} L_i, L_{min} = Min(L_1, L_2, ...L_N) \tag{5.1}$$

where N is the number of edges the the node connected to.

Half of this average length is the benchmark radius of the node. To prevent the generation of cracks where there is no deformation, we take the radius to be

$$r = (L_{avg} + L_{min})/2 \tag{5.2}$$

If any pair of those spheres collides, as before, a penalty term is added to the objective energy function.

$$E = E + c * min^2(0, d - r_0 - r_1) \tag{5.3}$$

where d is the distance between the centers of the two spheres, and r_0, r_1 are the radii of the two spheres.

In our second example, a projectile hits the center of a thin plate.

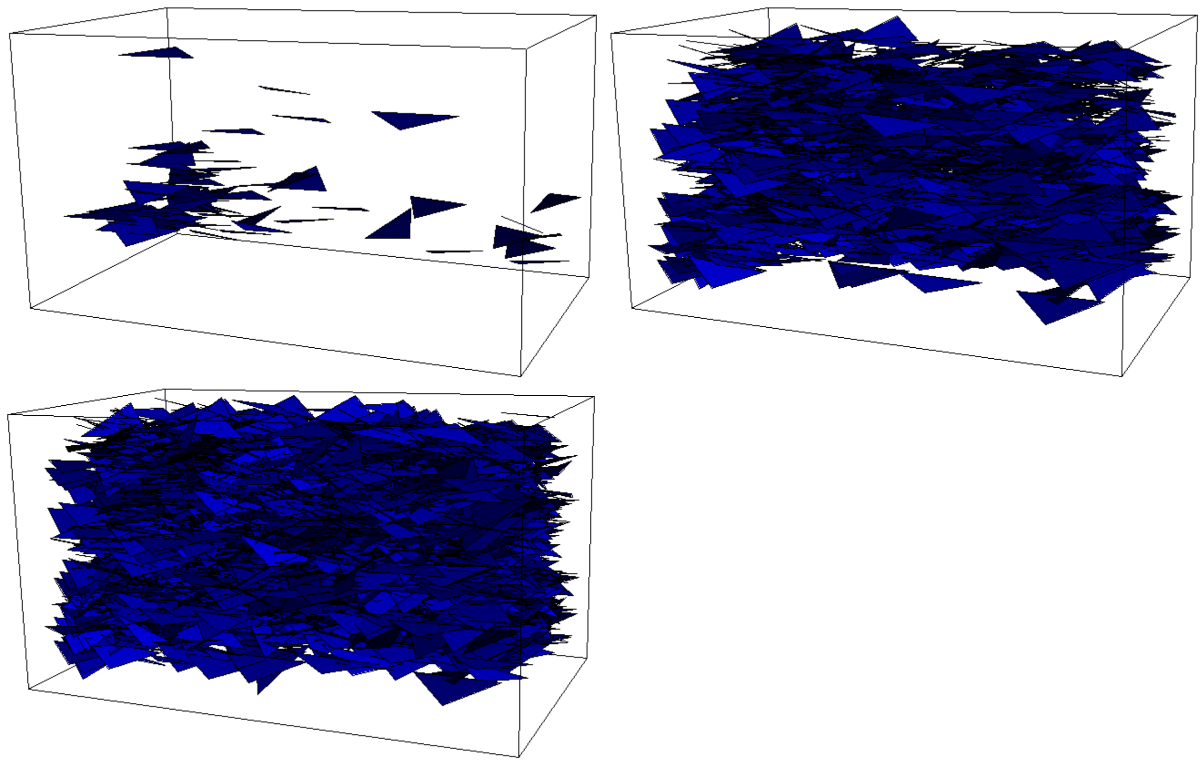


Figure 5.9: 3D cube pulled apart visualization by normal plane, result from one processor

The normal plane visualization is also from one processor. We can find that the approximation is not close enough, because the radial cracks and the comminuted region is not fully formed. In order to get a better solution, the best way is to use a accurate polyhedron collision detection and response algorithm. This will be completed in the future work.

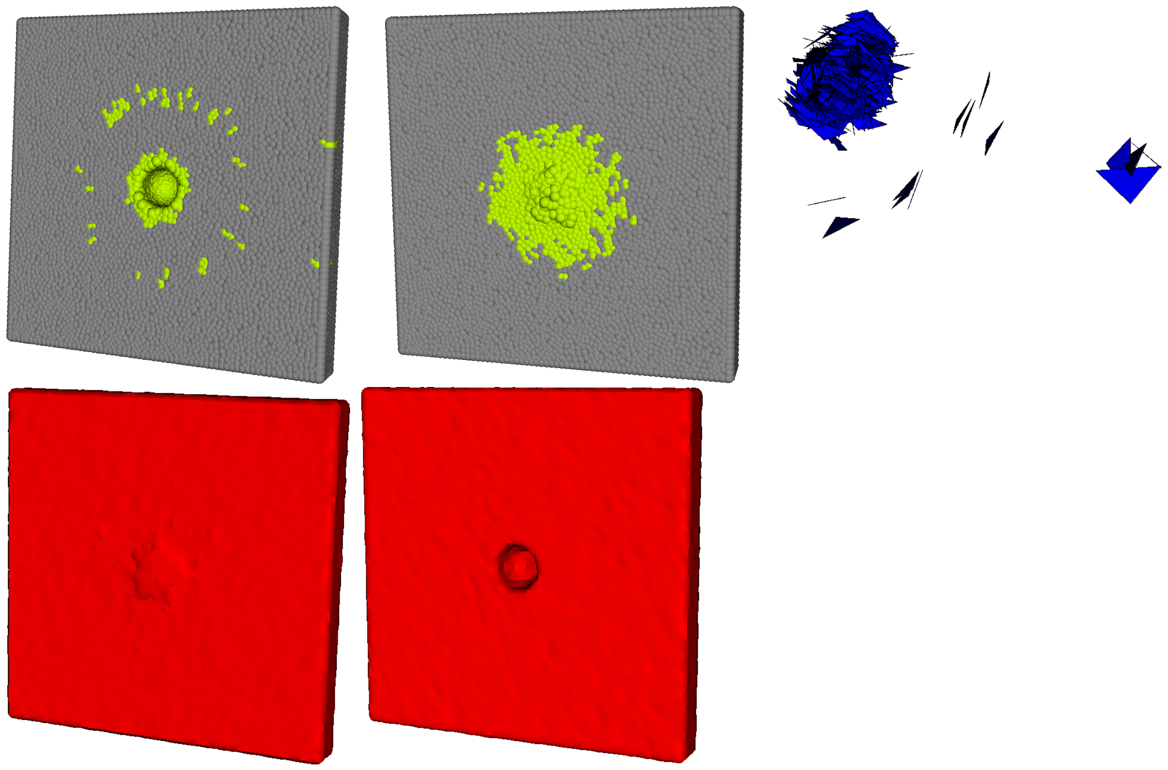


Figure 5.10: A 3D thin plate hit by a projectile in the center

5.5 Multiscale Computer Simulations of Fission Gas Discharge During Loss-of-Flow Accident in Sodium Fast Reactor

The purpose of this Section is to present the multiscale modeling approach for mechanistic three-dimensional transient computer simulations of the injection of a jet of gaseous fission products into a partially blocked SFR coolant channel following localized cladding overheat and breach. The described scenario is resolved on three different scales by inter-communicating computational multiphase fluid dynamics (CMFD) codes: FronTier, PHASTA and NPHASE-CMFD.

The smallest scale under consideration deals with the following processes: fuel rod stainless steel cladding failure, melting of the nuclear fuel and the subsequent fission gas discharge through the failure opening into the liquid sodium coolant. The characteristic size of this part of the problem is around 1 mm. Those processes are resolved using FronTier code. FronTier is a multiphysics code for the simulation of multiphase/free-surface flows based on the method of front tracking, which has been developed at Stony Brook University in collaboration with BNL and LANL. FronTier analyzes the shape of the cladding failure and provides the computed outflow rate of the fission gas to the PHASTA code.

The second scale of the simulation deals with the fission products being injected into the liquid sodium coolant. Using the geometry of the failure in the cladding and the gas injection flow rate a direct numerical simulation (DNS) of two-phase turbulent flow is performed by the two-phase version of PHASTA, combined with Level Set method. The characteristic size of this part of the problem is about 10 mm. PHASTA is a parallel, hierarchic (between 2nd- and 5th orders of accuracy, depending on function choice), adaptive, stabilized (finite element) transient analysis DNS flow solver (both incompressible and compressible). The turbulent two-phase PHASTA outflow is averaged over time to obtain mean phasic velocities and volumetric concentrations, as well as the liquid turbulent kinetic energy and turbulence dissipation rate, all of which serve as the input to the next scale of the simulations using the NPHASE-CMFD code. A sliding window time-averaging has been used to capture mean flow parameters for transient cases.

The largest scale of the simulation considers the flow of the liquid sodium coolant with fission gas in a reactor region around the failure with characteristic size of 0.5 m. This simulation is performed by NPHASE-CMFD code. NPHASE-CMFD is an advanced Computational Multiphase Fluid Dynamics computer program for the simulation and prediction of combined mass, momentum and energy transfer processes in a variety of multiphase/multiscale systems. It uses two-phase $k-\varepsilon$ models along with

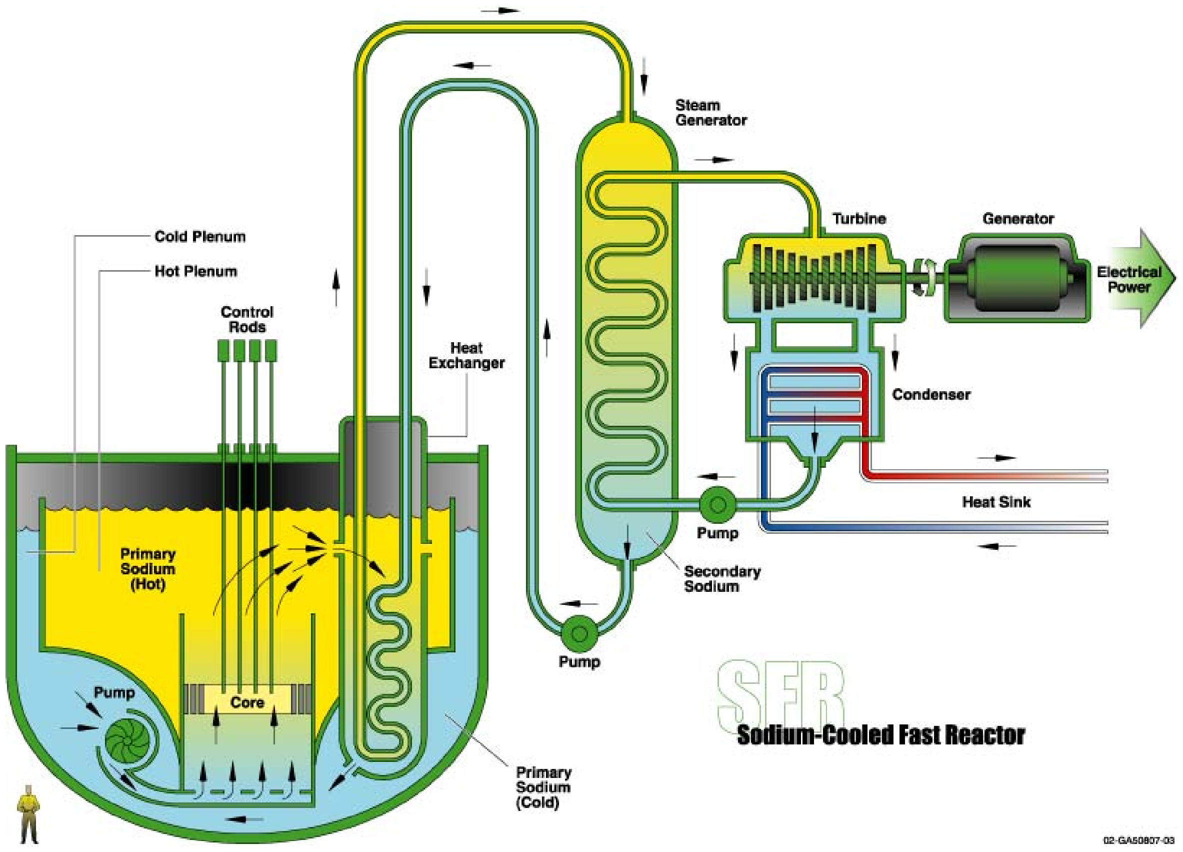


Figure 5.11: A general schematic of a Gen-IV Sodium-Cooled Fast Reactor (Source: A Technology Roadmap for Generation IV Nuclear Energy Systems by the U.S. DOE Nuclear Energy Research Advisory Committee and the Generation IV International Forum)

transient inflow boundary conditions supplied by PHASTA to perform turbulent flow simulation in the reactor coolant during the accident transient. NPHASE-CMFD also supplies the pressure value back to the PHASTA domain outflow. PHASTA, in turn, provides the pressure found at the inflow of its domain back to FronTier outflow. Thus, all the three codes used for the described multifield simulation are fully coupled. The use of such a multiple code platform allows one to perform full scale simulations of hypothetical accidents in future Gen-IV reactors (Fig. 42-43), while maintaining the required level of detail assured by the multiscale approach.

5.5.1 Description of individual computer codes

FronTier

FronTier is a computational package for direct numerical simulation of multiphase flows [65] developed at Stony Brook University in collaboration with LANL and BNL. It is based on front tracking [66], a numerical method in which surfaces of discontinuity are given explicit computational degrees of freedom. This method is ideal for problems in which discontinuities are an important feature, and especially where their accurate computation is difficult by other methods. FronTier supports compressible and incompressible Navier-Stokes equations and MHD equations in the low magnetic Reynolds number approximation [67], and phase transitions such as melting and vaporization [69]. The FronTier code has been used for large scale simulations of a variety of physical problems [67,68,70] on various platforms including the IBM BlueGene supercomputer New York Blue located at Brookhaven National Laboratory. The application of FronTier to the present problem has been two-fold. First, it has been used to model solid-state mechanics phenomena governing cladding heat-up and breach. Secondly, using the initial conditions of pressurized fission gas inside the reactor fuel pins prior to cladding failure, simulations have been performed of gas flow through the cracked cladding.

PHASTA

PHASTA is a parallel, hierarchic (higher order accurate from 2nd-5th order accurate depending on function choice), unstructured/adaptive, stabilized (finite element) transient analysis flow solver (both incompressible and compressible) [71,72]. PHASTA (and its predecessor, ENSA) was the first unstructured grid LES code [73] and has been applied to turbulent flows ranging from validation benchmarks (channel flow, decay of isotropic turbulence) to complex flows (airfoils at maximum lift, flow over a cavity, near lip jet engine flows and fin-tube heat exchangers). The PHASTA code uses advanced anisotropic adaptive algorithms and the most advanced LES/DES models [74]. Note that DES, LES, and DNS are computationally intensive even for single phase flows. The two-phase version of PHASTA utilizes the Level Set method to define the interface between the gas and liquid phases [75]. Here, it has been used to simulate gas jet injection into the coolant region using the boundary conditions provided by FronTier. Turbulent two phase PHASTA outflow is averaged over time to obtain mean velocities for each phase, turbulent kinetic energy and turbulence dissipation rate of each phase. This time-averaged data provides the input to the next scale of the simulation using the NPHASE-CMFD code. A sliding window time averaging has been used to capture slow changes to the mean flow parameters. Also, PHASTA provides the pressure found at the inflow of its domain back to FronTier outflow.

NPHASE-CMFD

NPHASE-CMFD is an advanced Computational Multiphase Fluid Dynamics computer program [79] for the simulation and prediction of combined mass, momentum and energy transfer processes in a variety of single-phase [76] and multiphase/multiscale systems, including gas/liquid, solid/liquid [77,78] and gas/solid/liquid [79] flows. It uses two-phase k-ϵ models of turbulence (the user can chose between the High Reynolds Number and Low Reynolds Number options of the model). In the present work, transient inflow boundary conditions have been supplied by PHASTA to simulate fission-gas/liquid sodium two-phase flow along and around the failed fuel element and the elements adjacent to it. NPHASE-CMFD also supplies the pressure value back to the PHASTA domain outflow. Thus, all the three codes used for the described multifield simulation are fully coupled.

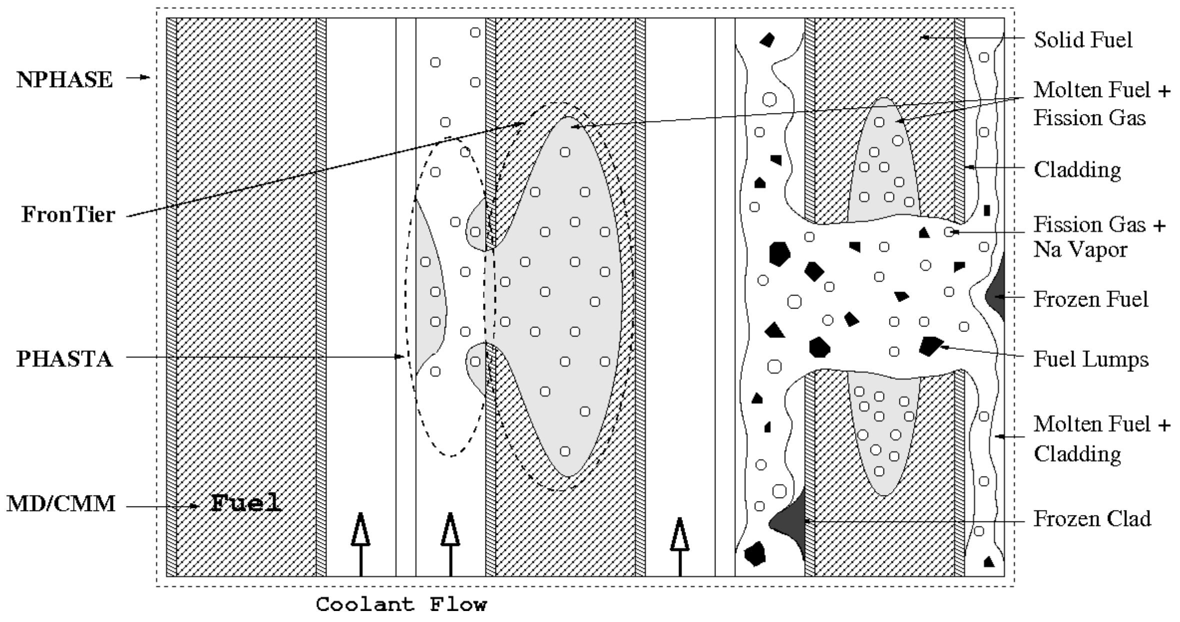


Figure 5.12: Schematic of fuel degradation and transport in SFR during fuel rod failure accidents

5.5.2 Plastic model simulation of fuel rod fracture

In this section, we illustrate the application of our algorithm to the study of accident scenarios in nuclear fuel rods. A nuclear fuel rod contains 8 mm diameter pellets of metallic or oxide uranium fuel in a stainless still cladding of 0.5 mm thickness. The total length of the fuel rod is about 2.5 m. A narrow (0.1 mm), irregular gap between the fuel and the cladding is used to transport fission gases from the fuel to the gas plenum on the top of the reactor core. Fuel rods are assembled in hexagonal structures and places into liquid sodium coolant that circulates in the reactor core and transports the thermal energy from fuel rods to the heat exchangers. The cross-section of the fuel rod is shown schematically in figure 5.13.

The temperature distribution across the fuel rod at normal operating conditions is shown in figure 5.15. The power production of about 60 kW/m in the fuel rod is balanced by the heat removal by the sodium coolant so that the temperature is in the steady state. The picture shows rapid changes of the temperature gradients across the

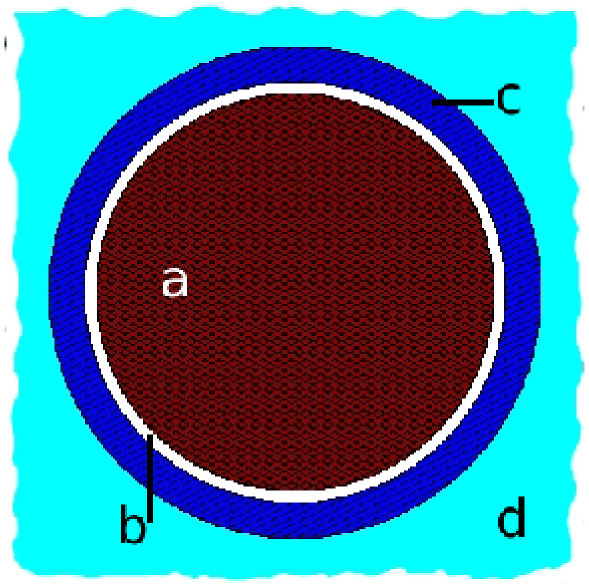


Figure 5.13: Schematic of the nuclear fuel rod cross-section: (a) is the fuel pellet, (b) is the gas gap, (c) is the stainless steel cladding, and (d) is the liquid sodium.

fuel rod due to sharp changes of the heat conductivity in the fuel, gap, and cladding. After years of operation, the surface of the fuel erodes and comes to a point-wise contact with the clad as shown in figure 5.14. The interior of the gas gap is not resolved on the mesh level and an empirical formula for the temperature jump across the gap [64] is applied for the heat transport calculations. For the related problem of the transport of fission gases in the gas gap, the flow in porous media equations are solved.

During either transient overheating or loss of coolant accidents, the heat transfer balance is changed and the temperature in the fuel rod increases. It can cause melting of the fuel rod and even stainless steel cladding. However because of the change of cladding properties with the temperature increase, the cladding usually fails before melting, causing the ejection of molten fuel and fission gases into the coolant reservoir. Figures 5.15 and 5.16 depict the increase of temperature and melting of the fuel in the transient overheating accident.

Using the solid fracture code, we have simulated the failure of the stainless steel

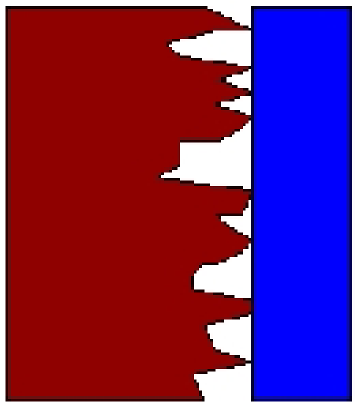


Figure 5.14: Shape of the gas gap between the fuel and cladding.

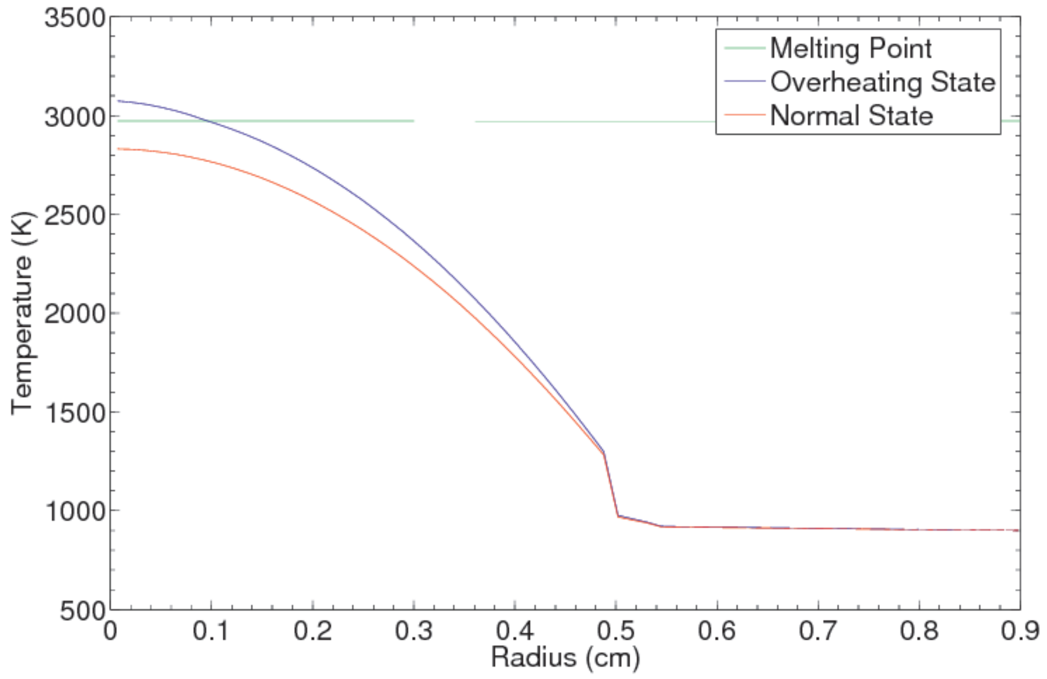


Figure 5.15: Temperature across the fuel rod at normal operating conditions and transient overheating.

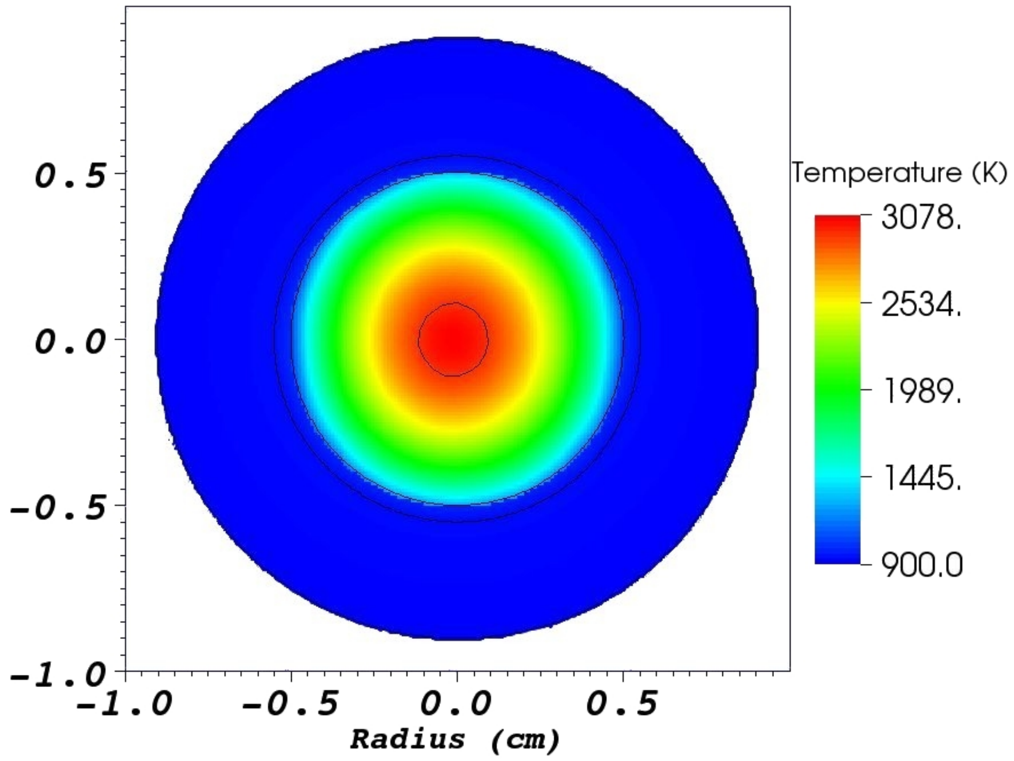


Figure 5.16: Temperature distribution across the fuel rod during transient overheating and the surface of the molten fuel.

cladding of the overheated nuclear fuel rod. In the present case, the cladding failed before melting, causing the ejection of fission gases into the coolant channels. The increasing fuel temperature caused the pressure inside the fuel rod to increase as well, thus augmenting the mechanical load on the cladding wall. The combined effects of elevated gas pressure and cladding temperature weaken the cladding wall and eventually lead crack formation. As no exact data on properties of the cladding material in a wide temperature range were available, our computational model is only qualitatively correct. We also employed a simplifies fluid-structure interaction model. Fluid-structure interaction is the interaction of some movable or deformable structure with an internal or surrounding fluid flow. The following picture is a sketch map of the basic mechanics of fluid-structure interaction. The connected line segments present the interface between the fluid and the solid, or the part lower than the line segments presents the fluid and the line segments presents the very thin solid shell whose thickness can be neglected. This is a constrained problem which we can take point as an example. is the projection of the area of and to tangent direction, and is the projection of composition of forces and to normal direction. is the average pressure to the area . According to Newton's third law, the constraints should be . The same idea is applicable to both 2D and 3D.

The exact model requires optimization with a several complex nonlinear constraints per each computational node. Such an optimization problem is very costly and the present research aims at the reduction of complexity of constraints without the reduction of the accuracy. The simplified model assumes that due to local change of properties as the materials enters the plastic deformation regime, the local deformation in the direction normal to the surface is a function of the material strength. Such an approach led to a micro-swelling of the fuel rod under the influence of pressure and temperature and the failure of cladding. The predicted crack side is shown in Fig.

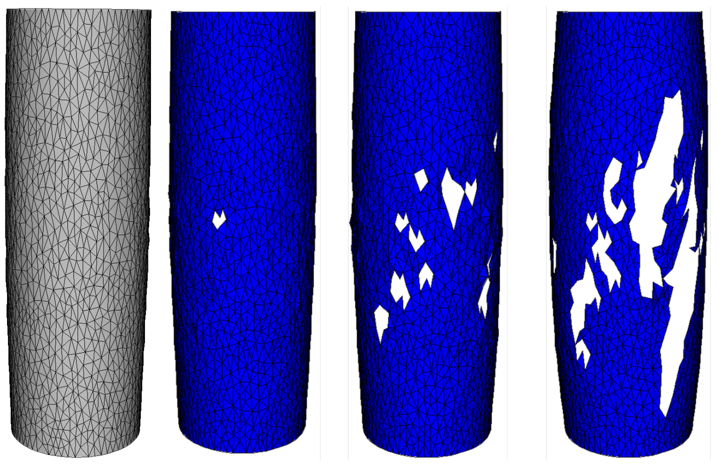


Figure 5.17: fuel rod modeling initial setup and fracture

5.17.

After the cracks are calculated, other packages will use the result to do other parts of the simulations in the project. For instance, FronTier will do the simulation of two-phase flow of the fuel and coolant.

Chapter 6

Conclusion

A new mass conservative mesoscale model for the simulation of fracture of solid materials has been developed. Our representation of solids by spring networks contains two degrees of freedom necessary to match real material properties and exhibits a stable Poisson ratio. The algorithm is based on the energy minimization of the network of triangular springs with critical strain and splitting of overstressed bonds and connected to them nodes ensuring the conservation of mass during the crack evolution. An algorithm to resolve the mesh folding and overlapping for the simulation of compressed materials has been developed by introducing special energy penalty terms. The main emphasis of the research is on the study of brittle fracture but elasto-plastic models for springs have also been developed for the simulation of plastic deformations with limited shear bands.

In 2D the mass is stored in mesh elements where the mesh in generated by Triangle. The method is material mass-conservative achieved by mesh splitting combined with overlapping prevention with approximately keeping the total area of the triangular mesh conservative. Due to the resolution of the mesh overlapping, the method can be applied to materials not only under tensile stress but also can deal with compression stress and explosive fragmentation simulation. The separated chunks of materials can freely move to any possible locations to form fragment flow if it agreed to the mechanical principals.

The model is also developed in 3D, where a particle method similar to smooth particle hydrodynamics is used. The mesh structure is standard unstructured tetrahedral mesh generated by TetGen, and the mass is stored in the particles which are the mesh nodes. Several preliminary simulations are presented, since the polyhedron collision detection and response are fully developed yet. Our software is implemented in parallel with different parallelization strategies. The first is the standard distributed parallelization with Global Arrays, which is a wrapper of standard MPI. The second is the parallelization with iterative Schwartz-type overlapping domain decompositions. During each iteration, each processor performs the optimization totally locally without data communication and the data is only been communicated after the iteration step is finished to update the boundary information of each subdomain. Due to the slow convergence of the overlapping domain decomposition, a number of sequence accelerating algorithms are analyzed.

One data visualization technique for 2D and three methods for 3D visualization are created. The 2D result is visualized by only displaying the crack surface and natural boundaries. In 3D, the three are a particle based visualization, iso-surface and trilinear interpolation based visualization and a normal plane visualization. Three different methods have their own advantages and disadvantages so that can be complementary to each other.

The software has been applied to the simulation of fracture of solids under slow stretching deformations, the rapid disintegration of highly tempered gases in the phenomenon called the Prince Rupert drop, and the fracture of thin brittle discs hit by high velocity projectiles. The bifurcation of the fracture dynamics from the growth of the comminuted zone to the propagation of isolated radial cracks, typical for the fracture of glass sheets and thin ceramic plates hit by projectiles, has been reproduced in out numerical experiments and scaling studies involving the change of material properties and projectile velocity have been performed. The fracture model has also been used in a coupled multiscale simulation of the nuclear fuel rod failure within a study of nuclear reactor safety issues. In the future, out can be applied to DOD fracture research of brittle materials and DOE nuclear reactor research. As steel and nuclear materials become brittle after sufficiently long exposure to the radiation of the level typical for nuclear rectors, the brittle fracture model may become a tool for the assessment of the reactor safety.

Bibliography

[1] A. A. Griffith, The phenomena of rupture and flow in solids, Philosophical Transactions of the Royal Society of London, A 221(1921): 163-198;

[2] G. Irwin, Analysis of stresses and strains near the end of a crack traversing a plate, Journal of Applied Mechanics, 24(1957), 361-364;

[3] T. A. Witten, L. M. Sander, Diffusion-Limited Aggregation, a Kinetic Critical Phenomenon, Phys. Rev. Lett. 47, 1400-1403 (1981);

[4] Leonard M. Sander, Fractal growth processes, Nature 322(1986), 789-793;

[5] Lincoln Paterson, Diffusion-Limited Aggregation and Two-Fluid Displacements in Porous Media, Phys. Rev. Lett. 52, 1621-1624 (1984)

[6] Maloy, K. J., Feder, J., Jossang, T. 1985. Viscous fingering fractals in porous media. Phys. Rev. Lett. 55:2688-91;

[7] J. Nittmann, G. Daccord, and H. E. Stanley, Fractal Growth of Viscous Fingers: A Quantitative Characterization of a Fluid Instability Phenomenon, Nature 314, 141-144 (1985);

[8] Ben-Jacob, E., Godbey, R., Goldenfeld, N., Koplik, J., Levine, H., Mueller, T., Sander, L., M.,Experimental Demonstration of the Role of Anisotropy in Interfacial Pattern Formation, Phys. Rev. Letts. Vol. 55, pp. 1315 (1985);

[9] Ball R C, On Growth and Form: Fractal and Non-fractal Patterns in Physics ed H E Stanley and N Ostrowsky (Dordrecht: Nijhoff) p 69(1986);

[10] Meakin P., Phase Transitions and Critical Phenomena vol 12, ed C Domb and J L Lebowitz (New York: Academic) p 336;

[11] Matsushita M, Honda K, Toyoki H, Hayakawa Y and Kendo H 1986 J. Phys. Soc. Japan 55 2618;

[12] Sander L M 1986 Nature 322 789;

[13] England A H 1971 Complex Variable Methods in Elasticity (New York: Wiley-Interscience);

[14] Landau L D and Lifshitz E M 1975 Theory of Elasticiry (Oxford: Pergamon)

[15] Mandelbrot B B 1982 The Fractal Geometry of Nature (San Francisco: Freeman)

[16] Witten T A and Sander L M 1981 Phys. Rev. Lett. 47 1400

[17] Meakin P., 1983a Phys. Rev. A 27 604

[18] Meakin P., 1983b Phys. Rev. A 27 1495

[19] Mandelbrot B B, Passoja D E and Paullay A J 1984 Nature 308 721

[20] Underwood E E and Banerji K 1986 Mater. Sci. Eng. 80 1

[21] Pande C S, Richards L R and Smith S 1987 J. M a w . Sci. Lett. 6 295

[22] Skjeltorp A T and Meakin P 1988 Nature 335 424

[23] Alder B J and Wainwright T E 1957 J. Chem. Phys. 27 1208

[24] Rahman A 1964 Phys. Rev. A 136 405

[25] Wood W W and Erpenbeck J J 1976 Ann. Rev. Phys. Chem. 27 319

[26] Jones R H and Gerberich W W (ed) 1985 Modeling Environmental Effects on Crack Growth Processes (Warrendale, PA: Metallurgical Society)

[27] Gehlen P C, Markworth A J and Kahn L R 1976 Computer Simulation for Materials Applications ed R J Arsenault, J R Beeler Jr and J A Simmons (Gaithersburg, MD: National Bureau of Standards)

[28] Shchukin D and Yushchenko V S 1981 J. Mater. Sci. 16 313

[29] Paskin A, Gohar A and Dienes G J 1980 Phys. Rev. Lett. 44 940

[30] Chakrabarti B K, Chowdhury D and Stauffer D 1986 Z. Phys. B 62 343

[31] M. A. Grinfeld, S. E. Schoenfeld, and T. W. Wright, Morphological instability of failure fronts, APPLIED PHYSICS LETTERS 88, 104102(2006);

[32] Paul Meakin, G Li, L M Sander, E Louis and F Guinea, A simple two-dimensional model for crack propagation, J. Phys. A: Math. Gen. 22 (1989) 1393- 1403;

[33] Paul D. Beale and David J. Srolovitz, Elastic fracture in random materials, Physical review B, Vol. 37, No. 10;

[34] Landau, Lifshitz, Theory of elasticity, 3rd edition, Elsevier, 1986;

[35] Landau, Lifshitz, Mechanics.3rd edition, Butterworth-Heinemann, 1976;

[36] Jonathan Richard Shewchuk, Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, in "Applied Computational Geometry: Towards Geometric Engineering" (Ming C. Lin and Dinesh Manocha, editors), volume 1148 of Lecture Notes in Computer Science, pages 203-222, Springer-Verlag, Berlin, May 1996;

[37] Allen Van Gelder, Approximate Simulation of Elastic Membranes by Triangulated Spring Meshes, Journal of Graphics Tools 3(2), 21-42(1998);

[38] V. Baudet, Modelisation et Simulation Parametrable d'objets Deformables, Applicaition Aux Traitements Des Cancers Pulmonaires, PhD dissertation, Universite Lyon I, June 2006;

[39] H. Delinette, Triangular Springs for Modeling Nonlinear Membranes, IEEE Transactions on Visualization and computer graphics, Vol. 14, No. 2, March/April 2008;

[40] P. P. Valko and J. Abate, Comparison of Sequence Accelerators for the Gaver Method of Numerical Laplace Transform Inversion, Computers and Mathematics with Applications 48 (2004) 629-536;

[41] Andrew H. Van Tuyl, Acceleration of convergence of a family of logarithmically convergent sequences, Mathematics of computation, Vol. 63, No. 207 July 1994, P. 229-246;

[42] J. P. Delahaye and B. Germain-Bonne, The Set of Logarithmically Convergent Sequences Cannot be Accelerated, SIAM Journal on Numerical Analysis Vol. 19, No. 4 (Aug., 1982), pp. 840-844

[43] I.A. Bolotnov1, F. Behafarid, D. Shaver1, S.P. Antal, K.E. Jansen, R. Samulyak , H. Wei and M.Z. Podowski, Multiscale Computer Simulation of Fission Gas Discharge during Loss-of-Flow Accident in Sodium Fast Reactor, OECD Nuclear Energy Agency IAEA Workshop(CFD4NRS-3), Experimental Validation and Application of CFD and CMFD Codes to Nuclear Reactor Safety Issues, Washington D.C., USA, 14-16 September 2010;

[44] S.A. Silling, E. Askari, A meshfree method based on the peridynamic model of solid mechanics, Computers and Structures 83 (2005) 1526-1535;

[45] S.A. Silling, Reformulation of elasticity theory for discontinuities and long-range forces, Journal of the Mechanics and Physics of Solids 48 (2000) 175-209;

[46] F. Bobaru, S. A. Silling, and H. Jiang, Peridynamic fracture and damage modeling of membranes and nanofiber networks;

[47] John B. Aidun and Stewart A. Silling, Accurate Prediction of dynamic fracture with Peridynamics, the Joint US-Russian Conference on Advances in Materials Science, Prague, 2009;

[48] M. Munawar Chaudhri, Self-Sustained Fracture Waves in Soda-Lime Glass, Materials Science Forum Vol. 662 (2011) pp 95-104;

[49] S. Chandrasekar M. M. Chaudhri, The explosive disintegration of Prince Rupert's drops, Philosophical Magazine Part B, Volume 70, Issue 6, 1994;

[50] Anthony Ball, On the bifurcation of cone cracks in glass plates, Philosophical Magazine A, 1996, Vol. 73, No. 4, 1093-1103;

[51] A. Ball and H. W. McKenzie, On the low velocity impact behaviour of glass plates, Journal de Physique IV, Colloque C8, supplement au Journal de Physique III, Volume 4, septembre 1994;

[52] TAO user manual, Argonn National Laboratory;

[53] Global Arrays user manual, Pacific Northwestern National Laboratory;

[54] OPT++ user mamual, Sandia National Laboratory;

[55] VisIt user mamual,, Lawrence Livermore National Laboratory;

[56] TetGen for 3D tetrahedral mesh, http://tetgen.org;

[57] Operations Research Models and Methods by Paul A. Jensen Jonathan F. Bard;

[58] S. Chandrasekar and M. M. Chaudhri, The explosive disintegration of prince rupert's drops, Philosophical Magazine B, 1994, Vol. 70, No. 6, 1195-1218;

[59] G. T. Camacho, M. Ortiz, Adaptive Lagrangian modelling of ballistic penetration of metallic targets, Comput. Methods Appl. Mech. Engrg. 142 (1997) 269-301;

[60] R. Brian Leavy, Rebecca M. Brannon, O. Erik, Strack, The use of sphere Indentation Experiments to Characterize Ceramic Damage Models, Int. J. Appl. Ceram. Technol., 7 [5] 606-615 (2010);

[61] Nicolas Moes, John Dolbow and Ted Belytschko, A finite element method for crack growth without remeshing, INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, 46, 131-150 (1999);

[62] Jeong-Hoon Song and Ted Belytschko, Cracking node method for dynamic fracture with finite elements, Int. J. Numer. Meth. Engng 2009; 77:360-385;

[63] A.R. Khoei, K. Karimi, An enriched-FEM model for simulation of localization phenomenon in Cosserat continuum theory, Computational Materials Science 44 (2008) 733-749;

[64] Duderstadt, J., Hamilon, L., Nuclear reactor analsis, John Wiley & Sons(1976);

[65] Du, J., Fix, B., Glimm, J., Li, X., Li, Y., Wu, L., A Simple Package for Front Tracking, J. Comp.; Phys., 213, 613-628 (2006);

[66] Glimm, J., Grove, J., Li, X., Tan, D.C., Robust computational algorithms for dynamic interface tracking in three dimensions, SIAM J. Sci. Comput. 21, 2240-2256 (2000);

[67] Samulyak, R., Du, J., Glimm, J., Xu, Z., A numerical algorithm for MHD of free surface flows at low magnetic Reynolds numbers:, J. Comp. Phys., 226, 1532-1549, (2007);

[68] Samulyak, R., Lu, T., Parks, P., A magneto-hydrodynamic simulation of pellet ablation in the electrostatic approximation, Nuclear Fusion, 47, 103-118 (2007);

[69] Wang, S., Samulyak, R., Guo, T., An embedded boundary method for parabolic problems with interfaces and application to multi-material systems with phase transitions, Acta Mathematica Scientia, 30B, 499-521 (2010);

[70] George, E., Glimm, J., Li, X., Li, Y., Liu, X., Influence of scale-breaking phenomena on turbulent mixing rates, Phys. Rev. E, 73, 0163041-5 (2006);

[71] Whiting, C.H. and Jansen, K.E., A Stabilized Finite Element Formulation For The Incompressible Navier-Stokes Equations Using A Hierarchical Basis, International Journal of Numerical Methods in Fluids, 35, 93-116 (2001).

[72] Jansen, K. E., Whiting, C.H. and Hulbert, G.M., A generalized-alpha method for integrating the filtered Navier-Stokes equations with a stabilized finite element method, Computer Methods in Applied Mechanics and Engineering, 190, 3-4, 305-319 (2000).

[73] Jansen, K.E., Unstructured Grid Large Eddy Simulations of Wall Bounded Flows, Annual Research Briefs, Center for Turbulence Research, NASA Ames/Stanford University, 151 (1993).

[74] Tejada-Martinez, A.E. and Jansen, K.E., A parameter- free dynamic subgrid-scale model for large- eddy simulation, Computer Methods in Applied Mechanics and Engineering, 194, No. 9, 1225-1248 (2005).

[75] Sethian, J. A., Level Set Methods and Fast Marching Methods, Cambridge University Press. (1999).

[76] Gallaway, T., Antal, S.P. and Podowski, M.Z., Multidimensional Model of Fluid Flow and Heat Transfer in Generation-IV Supercritical Water Reactors, Nuclear Engineering and Design, 238, 1909-1916 (2008).

[77] Tiwari, P., Antal, S.P. and Podowski, M.Z., Modeling Shear-induced Diffusion Force in Particulate Flows, Computers Fluids, 38, 727-737 (2009).

[78] Tiwari, P., Antal, S.P. and Podowski, M.Z., Three-Dimensional Fluid Mechanics of Particulate Two- Phase Flows in U-bend and Helical Conduits, Physics of Fluids, 18, 4, 1-18 (2006).

[79] Antal, S., Ettorre, S., Kunz, R. and Podowski, M., Development of a next generation computer code for the prediction of multicomponent multiphase flows, Proceeding of the International Meeting on Trends in Numerical and Physical Modeling for Industrial Multiphase Flow, Cargese, France (2000).